Laurea Specialistica in Informatica

presso la facoltà di Scienze Matematiche Fisiche e Naturali

Università degli studi di Pisa.

Confronto tra DBMS e Motori di Ricerca per Accesso a Dati XML, con Applicazioni

Relatore: Prof. Paolo Ferregina
Tutor Aziendale : Dott. Giuseppe Sisto

candidata: Maria Gabriella Ritondo

Indice

Introduzione ……..……6

Capitolo 1. Il progetto SPICE
1.1 Una panoramica ……..…….…….8
1.2 Il Consorzio ……..……..…….9
1.3 Struttura ospitante: Telecom Italia Lab …….……9
1.4 Motivazioni e sfide …….……..……13
1.5 Organizzazione del progetto …….……...……13
1.6 Il servizio di Content Guide …….……
1.7 Lo standard…….…….
1.7.1 Servizi di Content Guide: Contesto e normative di Riferimento

Capitolo 2. Realizzazione del servizio di Content Guide con DBMS
2.1 Architettura del Sistema
2.1.1 Interazione fra le Servlets
2.1.2 Interfacce e Funzioni interne
2.2 Definizione del Modello dei dati
2.2.1 Content Guide Data Base
2.2.2 Definizione dei messaggi XML

Capitolo 3. Realizzazione del servizio di Content Guide con Lucene
3.1 Il motore di ricerca Lucene
3.1.1 Una panoramica
3.2 DBMS vs Full-Text
3.3 Architettura del Sistema

Capitolo 4 . Lucene vs DBM
4.1 Creazione Repository
4.2 Prestazioni
4.3 Conclusioni

Appendice

Bibliografia

Dedicato a:

me stessa per non avere mai mollato,

mia madre per avermi sostenuto economicamente con tanti sacrifici,

Aureliano per avere sempre creduto in me, spronato e aiutato in tutti i modi possibili,

Loredana per essermi stata vicina e avermi spinto a prendere quel treno…,

e tutte le persone che hanno SEMPRE creduto in me.

Introduzione

SPICE (Piattaforma di Servizio per un Ambiente di Comunicazione Innovativo) è un progetto finanziato dall'Unione Europea e sviluppato dal consorzio omonimo che integra la competenza e la conoscenza dei principali operatori di telecomunicazioni europee e fornitori di servizi di IT.
Il consorzio SPICE è costituito da 23 partner di 11 paesi: 13 industriali di grandi dimensioni (operatori di telecomunicazioni, produttori di apparecchiature, sviluppatori piattaforma), 7 centri di ricerca e università, 2 PMI e una società di consulenza.
L'obiettivo di SPICE è la progettazione, sviluppo e rilascio di piattaforme di esecuzione di servizi di telefonia mobile efficienti e innovativi su reti 3G. Il progetto permetterà di supportare più piattaforme eterogenee e sarà utilizzata su domini di diversi operatori e in diversi paesi realizzando così una varietà di modelli di business.
Il Laboratorio di ricerca della Telecom Lab di Torino collabora a questo progetto sviluppando, in simbiosi con le altre aziende partecipanti, la parte inerente a nuovi servizi per la ricerca e il reperimento di file multimediali da rete mobile.
Nella presente tesi, si è sviluppato un confronto tra la scelta di utilizzare un DB da parte della Telecom e l'approccio al motore di ricerca Lucene suggerito dal mondo universitario per lo Storage e il Management delle informazioni.

Nel primo capitolo, viene data una panoramica su cosa è il progetto SPICE, quali sono i suoi obiettivi, l'organizzazione del progetto, la descrizione dei work-package che lo compongono e sullo sviluppo dell'ultima componente: la Content Guide.

Nel secondo capitolo, viene descritta la realizzazione del servizio di Content Guide con DBMS, le componenti Servlet che lo utilizzano, le loro interazioni temporali e i possibili messaggi XML che le componenti si scambiano per comunicare.

Nel terzo capitolo viene introdotto il motore di ricerca Lucene, facendo una panoramica sulla caratteristiche che lo compongono, le applicazioni che lo utilizzano come motore di ricerca e un breve confronto dello stato dell'arte con i DBMS.

Nel quarto capitolo vengono mostrati e discussi i test effettuati e le prestazioni raggiunte da entrambe le piattaforme per la creazione del repository e la sua interrogazione.

Una panoramica su SPICE

1.1 *Il progetto SPICE*

Il progetto SPICE (Piattaforma di Servizio per un Ambiente di Comunicazione Innovativo) è volto verso il problema ancora irrisolto della progettazione, sviluppo e rilascio di piattaforme di esecuzione di servizi di telefonia mobile efficienti e innovativi su reti 3G.
Il progetto tende verso la ricerca, la creazione di un prototipo e la valutazione di una sovra architettura estensibile e di un frame work di supporto, la creazione facile e veloce di servizi, e la distribuzione intelligente di comunicazioni mobili e dei servizi di informazione. Basandosi sui significativi progressi delle tecnologie informatiche, la piattaforma SPICE supporterà più piattaforme eterogenee consentendo l'esecuzione di nuovi servizi innovativi, e sarà utilizzata su domini di diversi operatori e in diversi paesi realizzando così una varietà di modelli di business .
Per gli utenti, gli operatori e i service provider, il progetto SPICE di oggi si trasformerà da disorganico ed eterogeneo in un punto facilmente gestibile e ricco di servizi d'ambiente sfruttando la diversità dei dispositivi e la capacità di promuovere l'adozione del servizio. L'approccio di SPICE amplierà le opportunità commerciali nelle comunicazioni e dei relativi settori di attività.
Per raggiungere questo ambizioso obiettivo, il consorzio di SPICE integra la competenza e la conoscenza dei principali operatori di telecomunicazioni europee e fornitori di servizi di IT nel progetto SPICE che fa parte di Wireless World Initiative (WWI).

1.1.1 Principali obiettivi di SPICE

I principali obiettivi del progetto SPICE sono:

- Fornire un modo facile e semplice per creare e installare servizi innovativi e per ridurre i tempi di sviluppo, costi e rischi ;
- creare un modo unico per fornire servizi su piattaforme eterogenee di esecuzione, terminali e reti ;
- Arricchire il panorama dei servizi, attraverso una struttura sovrapposta che supporti gli utenti ed offra un'esperienza personalizzata verso l'utente ovunque e in qualsiasi momento ;
- Crei una piattaforma fidata ed aperta che permetterà di semplificare l'utilizzo dei servizi, e dei dispositivi attraverso la personalizzazione ;
- Arricchire l'attuale piattaforma dei servizi con funzionalità di gestione dei contenuti e funzioni di servizio intelligenti controllata dal contesto di elaborazione delle informazioni ;
- Aprirsi a nuovi modelli e a catene commerciali ;
- Consentire un approvvigionamento del servizio europeo, pubblicizzando i servizi oltre i confini nazionali e commerciali ;
- Promuovere l'adozione di un innovativo software tecnologico nella piattaforma dei servizi d'ambiente delle telecomunicazioni.

1.2 Consorzio SPICE

Il consorzio SPICE è costituito da 23 partner di 11 paesi: 13 industriali di grandi dimensioni (operatori di telecomunicazioni, produttori di apparecchiature, sviluppatori piattaforma), 7 centri di ricerca e università, 2 PMI, una società di consulenza.

1.3 Struttura ospitante: Telecom Italia Lab

Nell'odierno mercato delle telecomunicazioni e dei nuovi media, "innovare per competere" è il concetto di fondo su cui si erige ormai il business e l'attività di tutti gli operatori. E tuttavia, utilizzare con profitto le nuove tecnologie implica un constante lavoro di ricerca, propedeutico e complementare ad ogni altra azione di mercato.
È in questo scenario che si colloca Telecom Italia Lab, la società del Gruppo Telecom Italia incentrata sulle attività di ricerca, studio e analisi; un centro di eccellenza attivo sulla scena italiana da oltre quarant'anni nei settori dello sviluppo di reti e servizi; una realtà che ha contribuito a raggiungere importanti traguardi come la definizione e il consolidamento del GSM, dell'MP3 e della trasmissione ottica.
I suoi esperti dislocati in numerosi laboratori, ciascuno afferente ad una specifica area di ricerca, operano per sviluppare innovazione e renderla rapidamente ed economicamente fruibile ai clienti del Gruppo, lavorando oggi all'implementazione delle reti d'accesso fissa e mobile, sviluppando servizi, piattaforme e sistemi (WiFi, UWB, WiMax, MoFi) e progettando terminali di nuova generazione.
Gli obiettivi del Centro di Ricerca si concentrano pertanto su alcune specifiche tematiche chiave: evoluzione della rete e dei servizi, evoluzione delle comunicazioni mobili, diffusione della larga banda, nuove soluzioni di identificazione e localizzazione.
In sinergia con le Università, i centri di ricerca e l'industria, Telecom Italia Lab opera quindi per progettare avanzati servizi mobili e multimediali, per il mercato corporate e consumer.
Così come si apprende sul sito istituzionale TelecomItaliaLab.com, il TILab nasce nel marzo 2001, dalla fusione di CSELT e della Business Unit Venture Capital di Telecom Italia, e a poche settimane dalla costituzione di Loquendo, la società in cui confluisce un settore di eccellenza di CSELT specializzato nello sviluppo di tecnologie e servizi vocali.
Oggi, le attività di Ricerca del Laboratorio si articolano in diverse sezioni, che sul sito dispongono ciascuno di uno specifico canale tematico.
Navigando in ciascuno di essi è possibile conoscere i progetti, le attività di laboratorio, le sperimentazioni e gli ultimi risultati nei diversi ambiti: High Speed Flexible Core Networks (la rete core fissa e mobile: evoluzione tecnologica e architetturale); Ubiquitous Seamless Access (reti d'accesso: dal doppino alla fibra, dal GSM verso la terza e la quarta generazione mobile); Enabling & Innovatine Services (servizi broadband e mobili, servizi di localizzazione e per la sicurezza, servizi Internet e multimediali); Terminals, Sensors, M2M (capacità funzionali evolute di terminali, sensori, sistemi M2M); Managing Complexity (la gestione di reti e servizi: supporto all'operatività e tecnologie di sistemi esperti e apprendimento).
I Laboratori si occupano delle architetture di rete: rete fissa tradizionale, reti ottiche trasmissive, reti mobili di ultima generazione, applicazioni innovative nei settori Internet e multimediale (IPV6, sistemi di codifica audio-video digitale, home e office networks, security, piattaforme IP, integrazione di sistemi, ecc.). Attraverso il lavoro congiunto con i laboratori di Torino e con il mondo delle Università in Italia e all'estero, si realizzano scenari pluriennali per valutare l'impatto dell'innovazione ICT sui mercati consumer e business, per poter poi proporre strategie di diffusione delle innovazioni. I settori oggetto d'indagine sono diversi: networking IP, comunicazioni mobili,

identificazione dei processi, interazione uomo-macchina, reti ottiche, router e reti IP, campi elettromagnetici, applicazioni Internet e multimedialità.
Di rilievo, infine le pubblicazioni realizzate, a carattere tecnico-scientifico, la linea editoriale sugli scenari del futuro per conoscere e promuovere l'evoluzione di metodologie e tecnologie applicate al settore delle telecomunicazioni, la rivista EXP per gli addetti al lavori.
Oggi continua a creare innovazione nei suoi laboratori progettando ed implementando la rete di accesso fissa e mobile, impegnandosi nell´evoluzione della rete di trasporto, sviluppando servizi e piattaforme, sperimentando e progettando terminali di nuova generazione; il tutto nell´attenta analisi delle esigenze del cliente finale e delle imprese che vedono nella rete di telecomunicazioni del futuro l´elemento abilitante per competere sul mercato a livello mondiale.

I laboratori sono ambienti tecnologicamente avanzati: circa 12000 mq di sofisticate e moderne infrastrutture, che spaziano dalla tecnologia di base agli studi sulle architetture di rete, dalla rete fissa tradizionale alle reti ottiche trasmissive. Dalle reti mobili di ultima generazione ad applicazioni innovative nei settori Internet e multimediale.

Tra le attività di ricerca in corso, che saranno percepite dal grande pubblico in modo particolarmente rilevante da qui a pochi anni, vanno citate: l´evoluzione delle comunicazioni mobili, dal cellulare di terza generazione e oltre a diversi sistemi che garantiscono un´alta velocità nell´accesso anche se in regime di condivisione di risorse (WiFi, UWB, WiMax, MoFi); la diffusione della larga banda, attraverso lo studio di modalità innovative per portare la fibra ottica fino a casa dei cliente; l´affermazione di nuove soluzioni di identificazione e localizzazione attraverso l´integrazione di funzionalità di telecomunicazione con tecnologie di tagging.
In stretto collegamento con università, centri di ricerca, e industria, Telecom Italia Lab avvicina il futuro con servizi avanzati in molti settori dal mobile al multimediale, per la casa e per l´impresa, garantendo qualità e sicurezza.
Il mercato italiano delle telecomunicazioni è considerato tra i più avanzati sia dal punto di vista tecnologico, sia nell'evoluzione delle attitudini e dei profili di consumo dei clienti.
L'innovazione tecnologica costituisce quindi per il Gruppo Telecom Italia, un **elemento essenziale** e differenziante **per sviluppare il proprio vantaggio competitivo** e mantenere la leadership in un mercato con livelli crescenti di competizione.
Il patrimonio di competenze tecnologiche e innovative del Gruppo ha consentito in questi anni la progettazione, lo sviluppo e l'adozione in campo di soluzioni di rete, di terminali e di servizi assolutamente all'avanguardia, patrimonio su cui far leva anche nei paesi esteri ove il Gruppo è presente con società controllate.
Le attività di innovazione tecnologica nell'ambito di Operations si concentrano sulle attività e sulle competenze rivolte **alla ricerca di base, alla valutazione delle tecnologie emergenti ed allo sviluppo "intra-moenia" anche presso le unità operative e di business** (Network, Market, Information Technology, Web & Media e Security).
Tra le attività in corso, che saranno percepite dal grande pubblico in modo particolarmente rilevante da qui a pochi anni, vanno citate: **l'evoluzione delle comunicazioni mobili** e la **diffusione della larga banda** in una visione di **evoluzione della rete e dei servizi** con un orizzonte temporale al 2015.
L'innovazione tecnologica del Gruppo Telecom Italia è inoltre il risultato di **partnership strategiche con i principali produttori di apparati** e sistemi per telecomunicazioni e **con centri di ricerca d'eccellenza** presso le più qualificate **istituzioni accademiche** nazionali ed internazionali (Politecnico di Torino e Milano, Università di Pisa, di Berkley, MIT..).
Più in dettaglio le attività di innovazione tecnologica vanno da interventi di **revisione delle tecnologie di base in una logica di aumento dell'efficienza nell'esercizio di rete e sistemi, fino a complesse attività di revisione radicale delle piattaforme, dei servizi e delle architetture**; essenziale è quindi l'impegno profuso sul campo dalle funzioni operative delle business unit per

assicurare **l'aderenza dei nuovi servizi alle esigenze del cliente** ed al continuo miglioramento dei livelli qualitativi di servizio.

1.3.1 Telecom: successi conseguiti

Nell'esercizio 2005 gli investimenti del Gruppo Telecom Italia relativi allo sviluppo ed **all'innovazione** ammontano complessivamente a **circa euro 3.700 milioni.**
Oggi in TILab - Innovation, Engineering, Testing operano 1600 ricercatori per sviluppare innovazione, ingegnerizzarla e renderla rapidamente ed economicamente fruibile ai clienti del Gruppo.
L'innovazione viene concepita, creata e sperimentata **nei laboratori di Torino e Roma,** (circa 12.000 mq) studiando la rete di accesso fissa e mobile, impegnandosi nell'evoluzione della rete di trasporto, sviluppando servizi e piattaforme.
Forte attenzione viene data all'opportunità di creare valore per il Gruppo Telecom Italia; tale attività viene perseguita attraverso una gestione strategica delle relazioni tra ricerca, Intellectual Property Right (IPR) e business finalizzata allo sviluppo del patrimonio brevettuale; in tale contesto, **nel 2005 sono stati depositati 82 nuovi brevetti**, **di cui 3** a seguito di progetti di ricerca congiunti **con i Pirelli Labs.**

I **principali risultati conseguiti nel corso del 2005** sono stati:

− rilascio del **servizio di Videotelefonia**, completamente basato **su** protocollo **IP** e sua interoperabilità con videocomunicazione mobile;
− rilascio del **servizio di Mobile Instant Messaging** (MIM) Blah sul mercato sud-americano;
− **arricchimento del servizio di Telecom Italia Alice Mia** grazie all'estensione al Personal Computer delle funzionalità oggi disponibili da telefono, consentendo così alla clientela sia chiamate di voce su IP che nuovi servizi telefonici supplementari e la gestione della qualità del servizio;
− diffusione di contenuti TV su terminale mobile attraverso sia la tecnologia UMTS che il **broadcasting DVBH**;
− **diffusione di contenuti musicali** a pagamento **su** terminale **mobile**: tramite la piattaforma di servizio i-Music Store;
− contribuzione al lancio del servizio **Alice Home TV**, tramite testing di laboratorio degli apparati di rete (DSLAM-IP e Piattaforma di Servizio) e del Set Top Box di utente e tramite conduzione di un trial di servizio con utenza amica.

Più in dettaglio **nell'ambito dell'Innovazione dei Terminali** sono da segnalare due importanti iniziative, entrambe tese ad incrementare i servizi fruibili da parte della clientela, assicurando nel contempo adeguati livelli di sicurezza:

− la prima è relativa all'integrazione nella **SIM** di un terminale mobile a tecnologia di prossimità **ZigBee:** grazie a questi lettori a RadioFrequenza e a basso costo facilmente integrabili all'interno dei terminali mobili, si permette la fruizione di nuovi servizi quali il tele-ticketing o il **pagamento automatico**;
− la seconda si riferisce allo sviluppo di soluzioni prototipali basate sull'utilizzo di **tecnologie biometriche** di riconoscimento del volto, che sono in grado di aggiungere sicurezza a servizi "sensibili" nella loro fruizione da terminale mobile dotato di telecamera: in questi contesti infatti

l’impronta del volto del cliente, memorizzata in parte su server di rete e in parte su SIM, viene confrontata con l’immagine rilevabile in tempo reale dal telefono, assicurando in questo modo un elevato grado di protezione da intrusioni indebite.

Nell’ambito dell’Innovazione della Rete Domestica si sono sviluppati:

− la costituzione della **Home Gateway Initiative** (HGI), il **nuovo Forum** che vede la partecipazione dei Vendor leader del settore tra cui Pirelli Broadband Solutions;
− le specifiche per **l’evoluzione del sistema di gestione della Home Network**, in modo da abilitare una gestione integrata degli apparati di rete domestica che porterà notevoli vantaggi sia a Telecom Italia nelle fasi di esercizio, che alla clientela finale nella percezione di qualità di servizio end-to-end fruito.

Nell’ambito dell’Innovazione della Rete geografica, sono stati forniti significativi contributi sia a livello architetturale che di specifici snodi tecnologici della rete, come dettagliato nel seguito:

− è stata delineata una **visione di lungo termine** per **una rete fissa-mobile** che, oltre a perseguire obiettivi di sinergie tecnologiche, abilita scenari di servizi fruibili tramite accessi fissi, mobili e multimodali nell’arco del prossimo decennio. Coerentemente con la strategia del Gruppo di abilitare **un’offerta Quadruple Play** (telefonia fissa, accesso Internet a larga banda, servizi su TV + comunicazione mobile) nei laboratori del Gruppo Telecom Italia si sono definite le specifiche tecniche dell’elemento di rete (QoS-Server) che garantirà al singolo cliente la “**Quality of Service”** necessaria per fruire il singolo servizio in modo ottimale;
− si è inoltre contribuito alla **stesura del Piano Tecnologico di Gruppo** relativamente ad aspetti di vision architetturale della rete per i prossimi anni;
− si è progettata e attivata la prima versione del **sistema di gestione multi-vendor** per l’attivazione di servizi innovativi basati **su DSLAM-IP**;
− si è sviluppata la **piattaforma IMS/IP Multimedia Subsystem**, con funzionalità in grado di trattare servizi di telecomunicazione avanzati basati su protocollo SIP/Session Initiation Protocol.

È da sottolineare infine che, **nell’ambito di Progetto Italia**, TILab - Innovation, Engineering, Testing **ha contribuito a diffondere l’immagine di Telecom Italia** come azienda innovativa con una serie di eventi tenutisi a Venezia (Convegno internazionale sulla Robotica, L’evoluzione delle Comunicazioni oltre l’orizzonte attuale, 4 passi nel futuro), a Genova (Festival della Scienza) e in varie altre manifestazioni in Italia e all’estero.

1.4 Motivazioni e sfide per Spice

Gli obiettivi di Spice sono volti al superamento di ostacoli affrontati nel momento delle creazione e consegna dei servizi mobili.
Esempi di tali ostacoli sono:

- il tempo troppo lungo per introdurre sul mercato il nuovo servizio ;
- i costi di sviluppo e integrazione troppo elevati a causa della complessità intrinseca e l'eterogeneità degli ambienti di esecuzione di servizio;
- gli utenti possiedono molti dispositivi di comunicazione differenti e sono circondati da molte tecnologie di accesso ma non possono trattare solitamente la complessità di accesso dei loro servizi attraverso alcuni di questi dispositivi;
- La fornitura di servizio coinvolge sempre più settori - Telco, fornitori di contenuti/servizi, terze parti come service provider e perfino utenti finali – che conduce all'aumento della complessità dell'ambiente per il funzionamento di servizio;
- La continuità di accesso del servizio da fisso a mobile e l'assenza di soluzione per i servizi di roaming tra operatori e la rete è ben lontano dall'essere una realtà ;

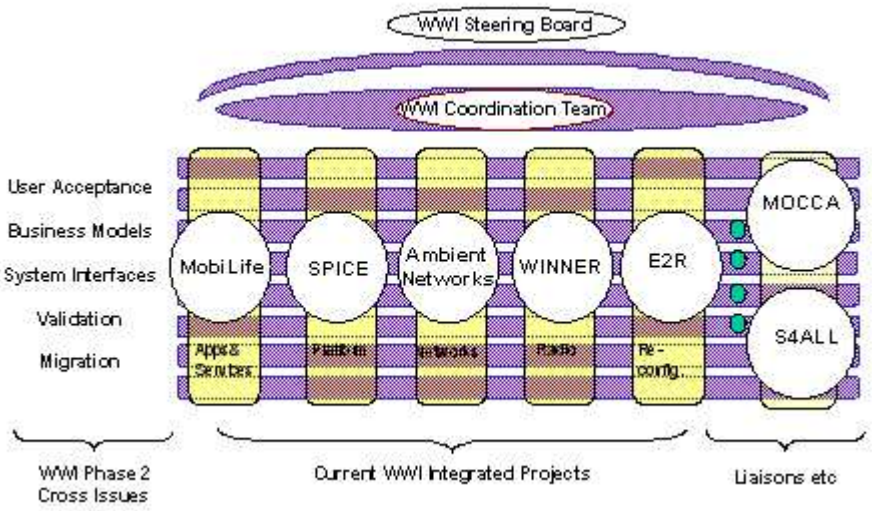
In questo contesto, le sfide principali per SPICE sono:

- Fornire all'utente finale i mezzi di comunicazione e le applicazioni adattate dovunque, in qualsiasi momento e su qualsiasi dispositivo;
- Fornire, ai service providers ed a gli utenti non professionisti, dei servizi abilitati che facilitano ed accelerino lo sviluppo di applicazioni;
- Consentire agli operatori di assumere il ruolo di Service Provider;
- Costruire un'infrastruttura trasparente per l'utente che nasconda la complessità dei servizi e delle applicazioni che viaggiano su diversi domini di accesso, in grado di gestire le varie tecnologie di rete di accesso ed offra una varietà di servizi.

1.5 Organizzazione del progetto

1.5.1 SPICE e WWI

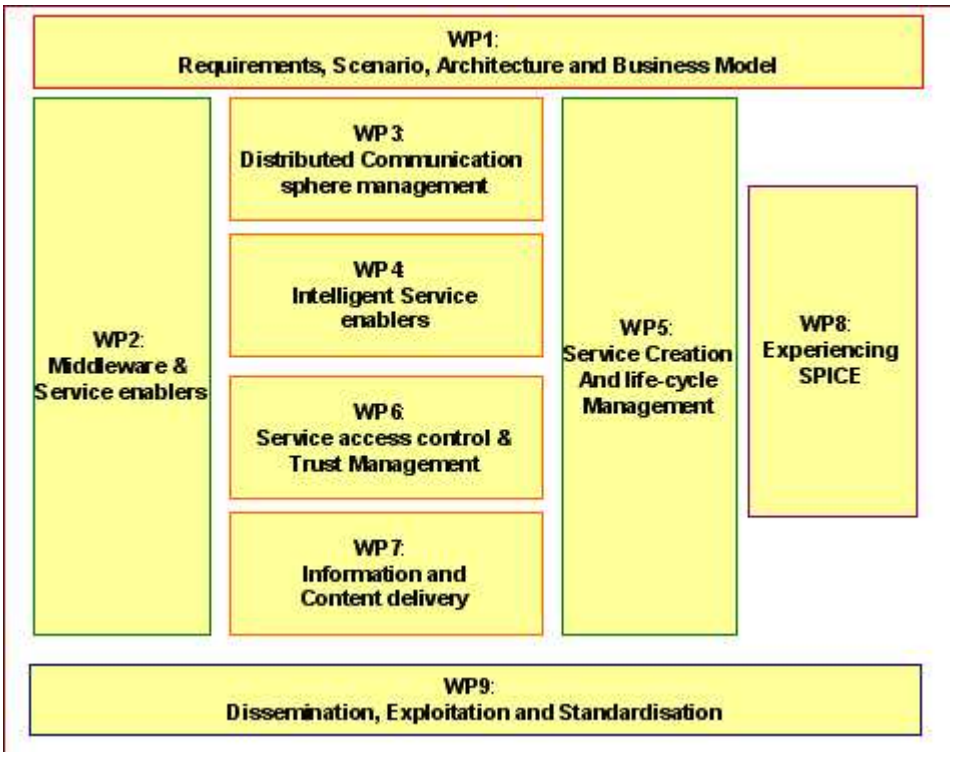
SPICE è coordinato con diversi progetti integrati, specialmente le Reti Ambientali e MobiLife, attraverso l'organizzazione Wireless World Initiative come mostrato di seguito.



WWI projects structure

1.5.2 Organizzazione del progetto e descrizione del Work-package

Il progetto Spice è strutturato in otto work-packages (WP) di lavoro, corrispondente a otto principali campi di ricerca. Altre due WP sono dedicati alla gestione e la diffusione del progetto, la standardizzazione e la formazione.



Work package structure of SPICE.

- **WP1** (Requirements, Scenarios, Architecture and Business Models) è focalizzato sull'architettura e il framework di SPICE . WP1 integra inoltre i risultati della R&S da parte degli altri WP in una coerente visione globale sulle comunicazioni mobili e sui servizi di informazione.

- **WP2** (Middleware & Service Enablers) mira a sviluppare un livello intermedio per garantire l'interoperabilità di lavoro fra le componenti di servizio distribuite, attraverso la messa in rete in vari settori e le imprese integrando la III parte dei service providers. I Sviluppatori di componenti utilizzeranno questa definizione di infrastrutture per rendere utilizzabili le componenti attraverso il servizio e la creazione di ambienti di esecuzione.

- **WP2** (& del middleware; I Enablers di servizio) punta sullo sviluppo di componenti del middleware per accertare l'interazione delle componenti distribuite di servizio, attraverso i vari dominii di impresa e della rete che integrano i fornitori di servizio dei terzi. Gli sviluppatori c useranno questa definizione dell'infrastruttura per rendere le componenti utilizzabili dagli ambienti della creazione e di esecuzione di servizio.

- **WP3** (Distributed Communication Sphere Management) mira a sostenere tecnicamente gli utenti con la creazione di meccanismi e soluzioni che gli forniscano un ambiente di comunicazionedi tipo 'always on' e il 'always best-configured'. L'obiettivo è anche quello di sfruttare il più possibile la diversità dei dispositivi (e le rispettive capacità) che costituiscono l'ambiente di comunicazione per l'utente.

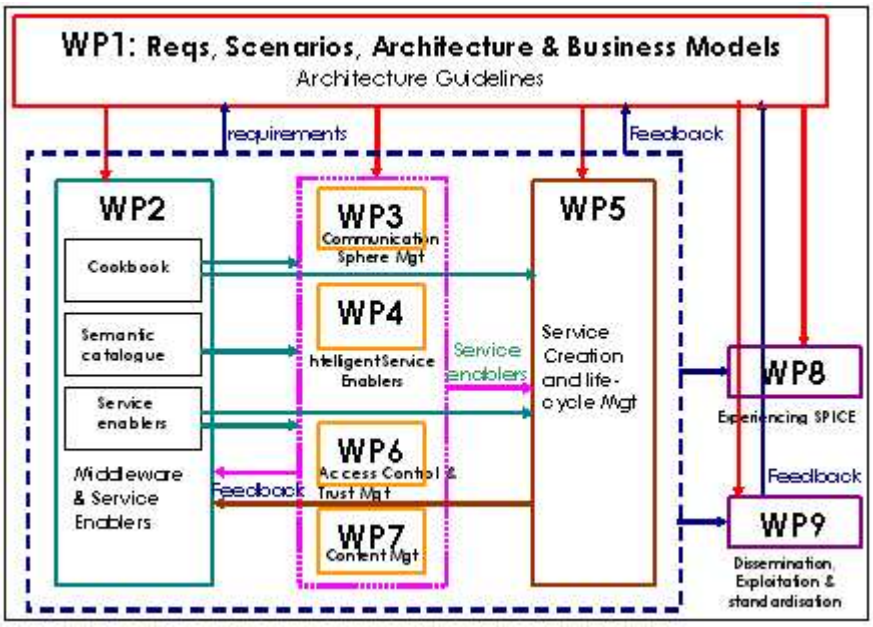
- **WP4** (Intelligent Service Enablers) mira a fornire soluzioni intelligenti per la piattaforma di servizio per il profilo utente e per il contesto di gestione delle informazioni e per l'adattamento pro-attivo del servizio (anticipatory and attentive middleware functionality).

- **WP5** (Service Creation and Life-Cycle management) specificherà e creerà il prototipo di un avanzato Service Creation e di un Execution environment che mireranno a fornire le tecnologie o al service provider con la capacità di progettare, sviluppare, consegnare ed eseguire rapidamente i nuovi servizi mobili per l'utente finale e i dispositivi, sulla base di componenti e servizi abilitanti e disponibili sulla piattaforma di servizi e sui dispositivi degli utenti.

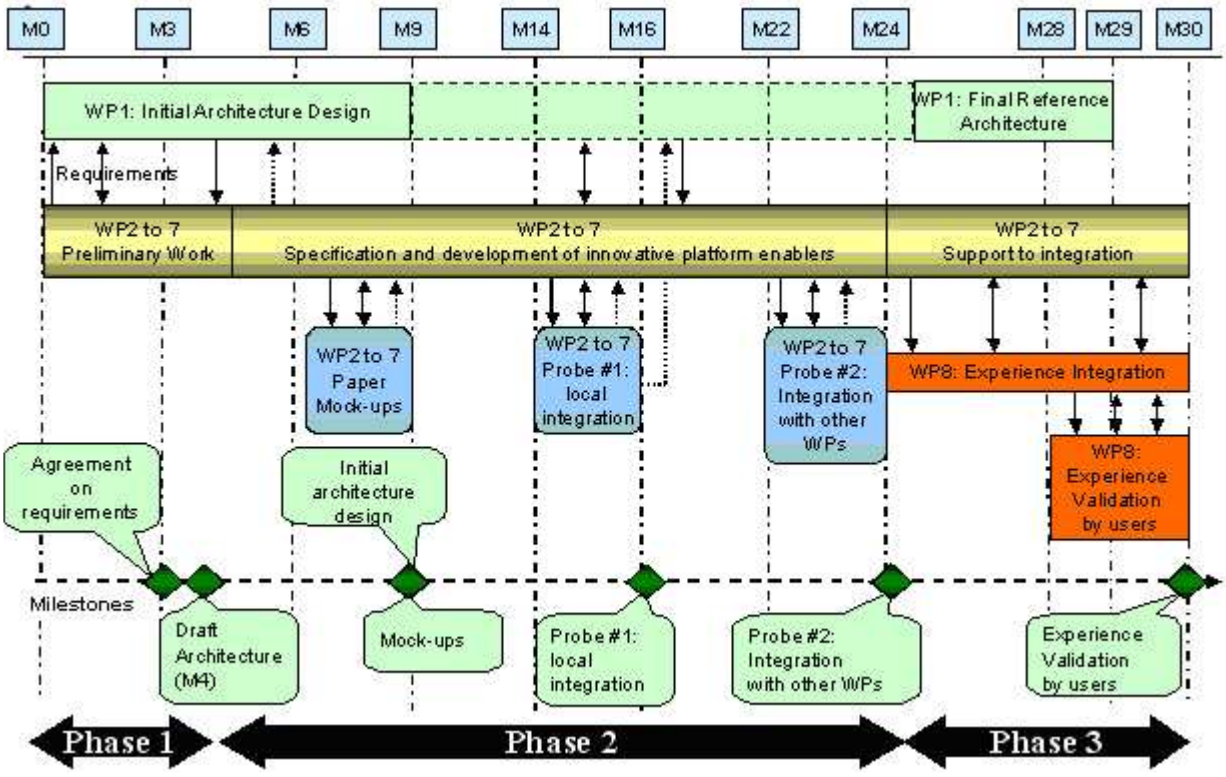
- **WP6** (Service Access Control and Trust Management) è incentrata su tutti gli aspetti relativi al controllo dell'accesso alla piattaforma di servizio per gli utenti e dei service providers. Questo include la fornitura di un framework di sicurezza a supporto degli utenti e dei servizi di autenticazione, autorizzazione, di non ripudio in ambiente con dominio singolo o multiplo e anche i metodi per la gestione e il rafforzamento dei livelli di servizio accordati.

- **WP7** (Content Management and Delivery): Questo WP si occupa principalmente della preparazione e fornitura di contenuti multimediali, e con il sostegno di informazioni che facilitano l'accesso a tali contenuti. Questi contenuti possono essere formattati in diversi modi per la consegna su varie reti a diversi dispositivi finali.

- **WP8** (Experiencing SPICE) ha lo scopo di assemblaggio e di convalida dei componenti di piattaforma in modo da dimostrare i risultati essenziali di SPICE. Gli scenari di dimostrazione mostreranno la capacità di operare su reti, ambienti e terminali eterogenei.



Organizzazione progetto SPICE e interazione fra i WPs.



Scadenze progetto.

1.6 Il servizio di Content Guide in SPICE

IL Work package 7 ha il compito di arricchire la piattaforma dei servizi di Spice con funzioni relative ai contenuti: a questo scopo il WP7 ha progettato un modo facile e semplice per distribuire i contenuti nel ambiente di servizio SPICE e una varietà di reti e di dispositivi per sostenere le funzionalità di preparazione, aggregazione, e consegna per contenuti multimediali.
Il WP7 gestisce direttamente anche i dati supplementari come i metadati dei contenuti e si basa su altri dati come ad esempio le informazioni relative al contesto e a gli utenti (in collaborazione con gli altri WP) per lasciare all'utente finale l'esperienza delle caratteristiche di SPICE sulla multi-modalità e sul pro-attivismo.
Dal significato di implementazione, i seguenti obiettivi chiave del WP7 sono tradotti in funzioni di lavoro:

- il servizio di Content Guide fornisce all'utente un livello uniforme, un accesso personalizzato relativo ai contenuti multimediali, in grado di sostenere la distribuzione di contenuti e di adattamento su piattaforme eterogenee, delle reti e dei terminali;
- l'integrazione della piattaforma di gestione dei contenuti con strumenti in grado di proteggere il contenuto e di meccanismi per ottenere l'accesso ai contenuti protetti (e.g. by intellectual property-related restrictions) a supporto del contesto eterogeneo e multi-dispositivo dell'utente;
- il sostegno del ruolo di utente finale come fornitore di contenuti nella piattaforma SPICE, rendendo i contenuti generati dall'utente finale disponibili nel Content Guide;
- consentire la distribuzione di contenuti sul Multi-device e consente l'interazione avanzata della sessione di mobilità e il rendering non monolitico del contenuto.

1 Realizzazione del servizio di Content Guide

1.1 Architettura di Sistema

L'implementazione del DB supporta pienamente il modello di dati del CGDM ed è notevolmente complesso e flessibile: ottenere tutte le informazioni necessarie per essere aggregate nel Content Guide (richiesto da e per essere visualizzata in vari Content Guide Client Panes) implica una comunicazione più complessa. Per esempio una "string search" dal client verranno trattati in più passi:

1. Una prima ricerca ritonerà gli IDs trovati
    - Per gli IDs corrispondenti ai contenuti trovati, le successive ricerche restituiranno come risultato le tabelle di SQL con i dettagliati dei contenuti.
2. Una successiva query può essere originata dal cliente per richiedere informazioni su un Bundle per uno specifico Content
    - Per gli IDs corrispondenti al Bundle trovato, le successive ricerche restituiranno come risultato le tabelle di SQL con i dettagliati dei content.

Il Query Agent (QA) è sviluppato come una servlet Java.
La comunicazione fra la servlet QA e la servlet CMS è sincrona, così come è sincrona la comunicazione fra il CMS e il Client.
Il QA è incaricato di implementare gli aspetti della sincronizzazione tra il CMS e il resto del sistema di Content Guide. Poiché il sistema dei metadati del DB deve essere accessibile dalla rete anche da altre servizi di SPICE (come il Recommender), che dovrà offrire una interfaccia di rete, attraverso una servlet Java "DBM" (Data Base Mediator).
La Servlet DBM si prenderà cura anche dei Metadati attraverso la funzionalità di Aggregazione (MA).

1.2 Servlets

1.2.1 Interazioni fra le Servlets

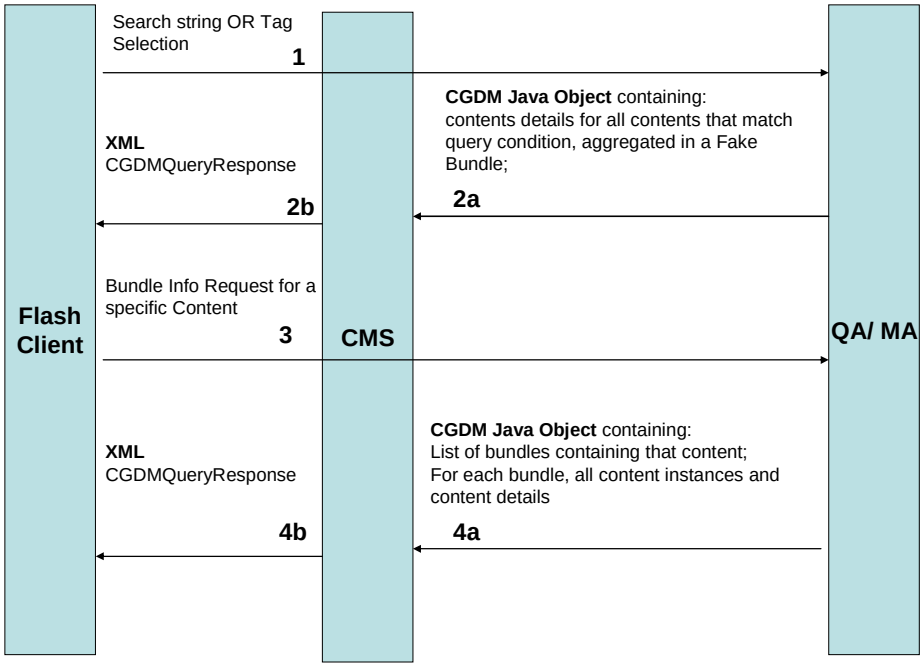


Tabella delle interazioni fra le componenti principali di SPICE.

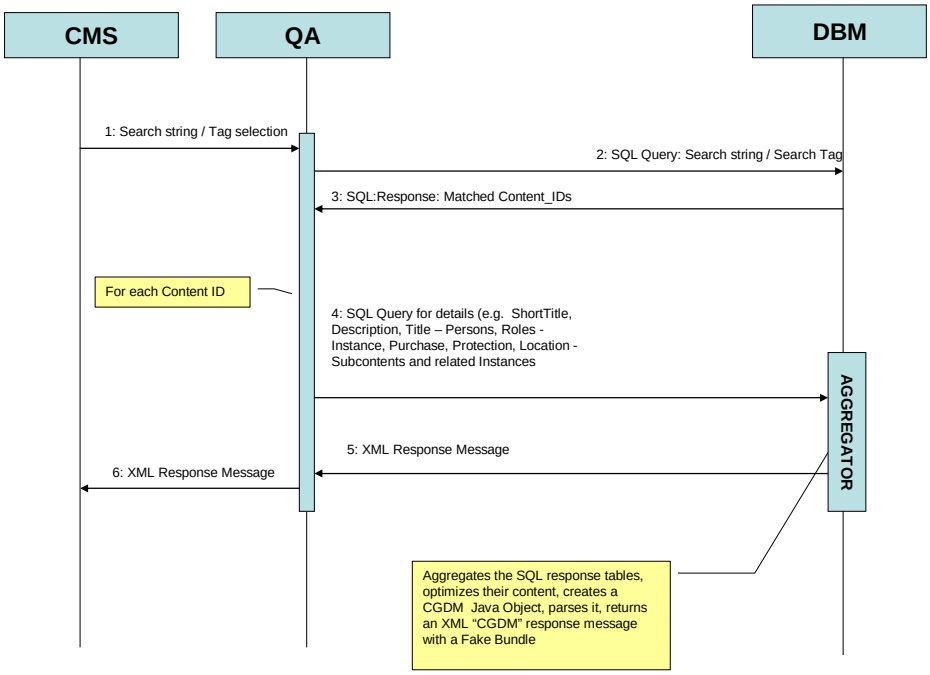


Diagramma temporale delle interazioni per la ricerca dei content's Id.

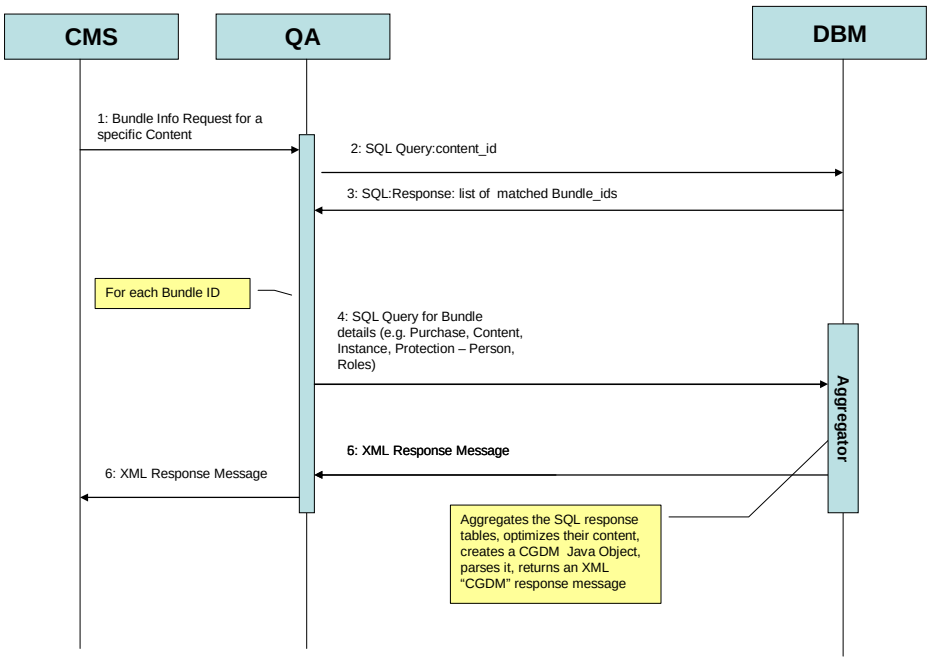


Diagramma temporale delle interazioni per la ricerca dei bundle's Id.

1.2.1.1 Content Mediator Service

3.1.3.1. Descrizione

Il Content Mediator Service è responsabile dell'esecuzione ed dell'adempimento delle richieste dell'utente

The Content Mediator Service is in charge of processing and fulfilling user requests coming from the client application (flash client), providing the content guide metadata to the client application, managing the selection of contents, informing the Learning system about the user content selections, communicating with the profile manager to retrieve the end user content guide profile and communicating with the Charging and Billing module. More specifically, the Content Mediator Service provides the intelligence to play the role of mediator between the client and the Query Agent within the Content Selection Decision point. The CMS module is implemented as a Java servlet.

CMS interfaces are detailed in Appendix A par. 3.1.2.1.

1.2.2 Content Selection Decision Point

Description

The Content Selection Decision Point represents just a logical wrapper for the enablers Query Agent and Metadata Aggregator do not correspond to any individual SW component. QA and MA are in charge of searching, managing, selecting and aggregating the various inputs that contribute to population of the Content Guide. The synchronization aspects are managed by the QA.

1.2.3 Query Agent (QA)

Description

The QA is implemented as a Java Http Servlet "*QAServlet*". It is managed by Servlet Container that specifies a runtime environment, and takes care of security, concurrency, life-cycle management, transaction, deployment, and other services.
The QA is in charge of the synchronization aspects between the the CMS and the other components of the Content Guide system.
The QA receives XMLRequests from CMS, performs internal parsing and creates the SQL Queries to be forwarded to the DBM.
The QA receives the XML Message containing the Content Guide and returns it to the CMS.
QA interfaces are detailed in Appendix A par. 3.1.2.2.

1.2.4 Metadata Aggregator (MA)

Description

MA is implemented as a function within the DBM Servlet (see par. 3.1.7). It has the following functionalities:

- Parsing the SQL response tables received from DMB and aggregating them, on the basis of the synch info received from the QA
- Transforming the aggregated tables into a CGDM XML Message representing the Content Guide and returning them to the QA.

MIS Interface to Metadata Repository

This interface offers a query interface to the Metadata Repository over the network.
It will be implemented as Java servlet *DBM*, as indicated in **Errore. L'origine riferimento non è stata trovata.**.

Description

The DBM is implemented as a Java Http servlet. It offers the query interface to the Metadata Repository over the network and implements Metadata Aggregator functionalities.
It receives the SQL Queries from the QA and executes them against the Metadata Repository. Query Results are parsed, aggregated and returned to the QA, in the form of an XML response message, compliant with the CGDM Data Model.
DBM interfaces are detailed in Appendix A par. 3.1.2.3.

3.1.2 Interfacce e Funzioni

3.1.2.1 Content Mediator Service

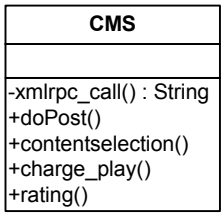


Figure 1: C7.1.3 – Content Mediator Service

Funzioni Interne

Method xmlrpc_call()
Parameters: String (uri), String (resource name), String (xml input)
Returns: String (whole xml response composed of aggregated metadata)

- xmlrpc_call: comunica con il componente T7.3 responsabile della riproduzione del contenuto, ilservizio dell'utente finale. La presente comunicazione viene eseguito inviando il messaggio XML a un URI utilizzando Apache XML-RPC.

Method doPost()
Parameters: HttpServletRequest (request), HttpServletResponse (response)
Returns: void

- doPost: Questo metodo è il cervello della nostra servlet, esegue la parte principale della funzionalità del CMS. Questo (come un servlet) è responsabile per settare il type, il codice di errore, i cookies e le altre informazioni di stato sulla richiesta (in questo caso, dal Flash client) in modo da consegnare i risultati delle interrogazioni. Questo metodo è invocato da un messaggio di POST. In aggiunta, riceve e parifica la richiesta del Client(ContentSelection, Play, Feedback o altri tipi di messaggi del client)decidendo quale componente è responsabile per ogni richiesta.

3.1.2.1.1 ***Method contentselection()*** *(see Section 4.1.2. CMS – Charging& Billing)*

3.1.2.1.2 ***Method charge_play()*** *(see Section 4.1.2. CMS – Charging& Billing)*

3.1.2.1.3 ***Method rating()*** *(see Section 4.1.3. CMS – Learner)*

3.1.2.2 Query Agent (QA)

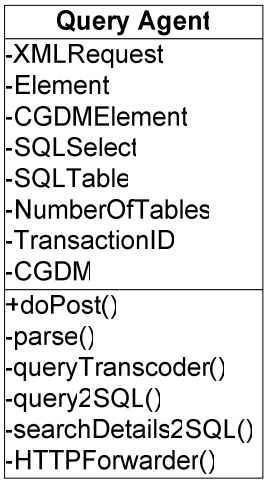


Figure 2: C7.1.2.1 – Query Agent

Interfacce

Method: doPost ()
Parameters: HttpServletRequest (req), HttpServletResponse (resp)
Returns: void
Throws: ServletException

- doPost: Questo metodo è il cervello della nostra servlet, esegue la parte principale della funzionalità del CMS. Questo (come un servlet) è responsabile per settare il type, il codice di errore, i cookies e le altre informazioni di stato sulla richiesta (in questo caso, dal Flash client) in modo da consegnare i risultati delle interrogazioni. Questo metodo è invocato da un messaggio di POST.
- executes the main part of the QA functionalities; it is responsible for setting the mime type, error code, cookies and other status information about the request, as well as delivering the results of the requests. This method is invoked by a POST message.

Funzioni Interne

Method: parse ()
Parameters: String (XMLRequest)
Returns: Tree (Element)

- Descrzione: riceve una richiesta XML in input dal CMS, la parsifica e la trasforma in un albero di elementi.

Input Parameters:

Name	Type	Description
XMLRequest	String	Document representing XML request (from CMS)

Output Parameters:

Name	Type	Description
Element	Tree	Tree of Elements

Method: queryTranscoder ()
Parameters: Tree (Element)

Returns: Tree (CDGMElement)

- Descrizione: legge l'albero degli elementi, crea l'albero dei CGDMElement e setta I relative attribute (Type, Name, Value) per ogni nodo.

Input Parameters:

Name	Type	Description
Element	Tree	Tree of Elements

Output Parameters:

Name	Type	Description
CGDMElement	Tree	CGDM Tree of Elements

Method: query2SQL ()
Parameters: Tree (CGDMElement)
Returns: string (sqlSelect)

- Descrizione: legge l'albero dei CGDMElement e I relative attribute (Type, Name, Value) per ogni nodo, crea la queri SQL che deve essere inviata al DBM. In particolare la classe Query2SQL ha diversi metodi interni; i più importanti sono elencati di seguito:

Method: querySQL ()
Parameters: Tree (CGDMElement)
Returns: string (SQLSelect)

- Descrizione:legge l'albero degli elementi cgdm e I relative Attributi (Type, Name, Value) per ogni nodo, crea la query SQL da inviare al DBM.

I seguenti metodi corrispondono al tipo di interazione scelta dal Client:

Method: tagCloud ()
Parameters: -
Returns: String (SQLSelect)

Method: tagSelection ()
Parameters: Tree (CGDMElement)
Returns: String (SQLSelect)

Method: search ()
Parameters: CGDMElement
Returns: String (SQLSelect)

Method: readTree ()
Parameters: CGDMElement
Returns: String (SQLSelect)

Method: sqlParameterType ()
Parameters: CGDMElement
Returns: String (SQLSelect)

Input parameters:

Parameter Name	Parameter Type	Description
CGDMElement	Tree	CGDM Tree of Elements

Output parameters:

Parameter Name	Parameter Type	Description
SQLSelect	String (SQL)	compliant with SQL syntax, with specific query parameter (SELECT, FROM WHERE), depending on the case

Method: searchDetails2SQL ()
Parameters: SQLTable
Returns: string (sqlSelect)
Dascrizione: legge le tabelle SQL che gli sono ritornate dal Query2SQL (contenenti la lista dei Content_IDs oppure Bundle:IDs), e crea le query SQL (per leggere I dettagli dei Content/Bundle) da inviare al DBM.

Input parameters:

Parameter Name	Parameter Type	Description
SQLTable	SQL Tables	CGDM Tree of Elements

Output parameters:

Parameter Name	Parameter Type	Description
SQLSelect	String (SQL)	compliant with SQL syntax, with specific query parameter (SELECT, FROM WHERE), depending on the case

Method HTTPForwarder()
Parameters: String (URL_< destination>), String (xml input)
Returns: String

- Description: HTTPForwarder gestisce la comunicazione con la servlet DBM. La presente comunicazione è eseguita inviando un messaggio XML su HTTP.

3.1.2.3 Multimedia Inventory Service (MIS to Metadata Repository)

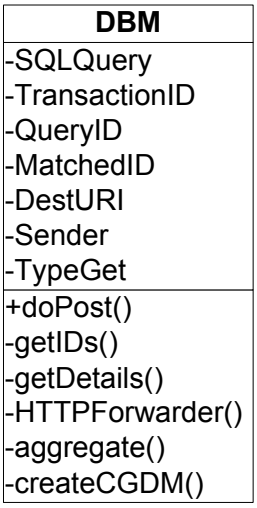


Figure 3: C7.1.1 – DBM Servlet -- Multimedia Inventory Service

Interfacce

Method: doPost ()
Parameters: HttpServletRequest (request), HttpServletResponse (response)
Returns: void

- doPost: Questo metodo esegue la parte principale delle funzionalità del DBM. E' responsabile del settaggio del type, del codice di errore, dei cookies e di altre informazioni sullo stato della richiesta, così come la consegna dei risultati della richiesta. Questo metodo è invocato attraverso un messaggio di POST.

Internal Functions

Method: getIDs ()

Parameters: SQL (SQLQuery), Integer (TransactionID), Integer (QueryID), String (DestURI)
Returns: SQLTable (MatchedIDs)

Input Parameters:

Name	Type	Description
SQLQuery	SQL	specific query to search content_IDs or Bundle_IDs
TransactionID	Integer	Transaction Identifier
QueryID	Integer	Query Identifier
DestURI	String	Address to which the response has to be returned (QA)

Output Parameters:

Name	Type	Description
MatchedIDs	SQL Table	List of Matched Content/Bundle IDs -- returned to QA

Method: getDetails ()

Parameters: SQL (SQLQuery), Integer (TransactionID), Integer (queryID), Integer (MatchedID), String (DestURI)
Returns: SQLTables

Input Parameters:

Name	Type	Description
SQLQuery	SQL	specific query to get content/bundles details
TransactionID	Integer	Transaction Identifier
QueryID	Integer	Query Identifier
ID	Integer	Content or Bundle Identifier
DestURI	String	Address to which the response has to be returned (QA)

Output Parameters:

Name	Type	Description
SQLTables	SQL Table	Matched Content/Bundle Tables

Method: aggregate ()

Parameters: Integer (TransactionID), Integer (NumberOfTables), Integer (QueryID), SQLTables
Returns: SqlTable (AggrSqlTable)

- Description: receives (sorted) SQL tables in input, aggregates them

Input Parameters

Name	Type	Description
TransactionID	Integer	Transaction Identifier (input from DBM and QA)
QueryID	Integer	Query Identifier (input from DBM)
NumberOfTables	Integer	Input from QA
SQLTables	SQL Table	Input sorted SQL Tables

Output Parameters:

Name	Type	Description
AggrSQLTable	SQL Table	Aggregated SQL Table)

Method: createCGDM ()

Parameters: AggrSQLTable
Returns: String (CGDM)

- Description: receives aggregated SQL tables in input;

Input Parameters:

Name	Type	Description
AggrSQLTable	SQL Table	Aggregated SQL Table

Output Parameters:

Name	Type	Description
CGDM	XML String	CGDM XML Message --returned to QA

3.1.2.2 Query Agent Servlet

- **Servlet Invocation**

  res = HTTPForwarder.doPost(<*URL_QAServlet*>, <*npv*>)

  npv is a message (in the form of a vector) that contains the input parameters expected from the CMS (see below)

- **Input Parameters (from CMS)**

  - (String) **XmlRequest** -- *XML Query Message*
  - (String) **QueryID** – *assigned by the CMS*
  - (String) **Sender** -- *expected value: CMS*

  Note: at the moment the following input XML Query Messages from CMS are taken into account:

  - Method= *TagCloudReq*
  - Method= *TagSelection* / Type= *TAG*
  - Method= *Search* / Type= *SearchString*
  - Method= *Search* / Type= *FieldSelection* / Name= *Classification*
  - Method= *SearchBundles* / Name= *Content_ID*
  - Method= *MostViewed*
  - Method= *UpdateMostViewed*/ *Name= Content_ID*

- **Output returned to CMS**

  (String) **XML Response Message**

3.1 QA Servlet behaviour description.

1. The QA Servlet is called by the CMS; depending on the input message, it extracts the query and transforms it into a *CGDMElement Tree* (through function *Parse* and *QueryTranscoder*).

Then (through *Query2Sql*) the SQL Queries are created from CGDMElement Tree, to be forwarded to the DBM.

2. A first interaction with the DBM takes place

   Depending on the case:

   - XML Query Message: Method: *TagCloudReq*
     - A **search for Tag Could** is performed

   - XML Query Message:
     Method= *TagSelection* / Type= *TAG*
     Method= *MostViewed*
     Method= *ContentDetails* / Name= *Content_ID*
     Method= *Search* / Type= *SearchString;*
     Method= *Search* / Type= *FieldSelection* / Name= *Classification*
     - A **search for ContentIDs** is performed

   - XML Query Message:
     Method= *SearchBundles* / Name= *Content_ID*
     - A **search for BundleIDs** is performed

   - XML Query Message:
     Method= *UpdateMostViewed* / Name= *Content_ID*
     - An **update** of field n_views in table content is performed

3. The DBM Servlet returns to the QA

   - in the case of **TagCloud:** an **XML CGDM**, which is sent by the QA to the CMS without no other manipulation

   - in the case of **Update**: an update notification with no other interaction

   - in the case of **Content**
     Methods *TagSelection, MostViewed, Search*: a **string containing** Content **IDs**
     Method *ContentDetails*: no interaction needed, as the list of Content_Ids is already available

   - in the case of **Bundle** (method *SearchBundle*)**:** a **string containing** Bundles **IDs**

4. In the case of the IDs, a second interaction with the DBM has to take place. The QA Servlet invokes *SearchDetails* to create the actual SQL queries. Depending on the case:

   - In the case of **Content, 3 queries** will be created:

     - *Query1* to search *Title*, *Description*, *ShortTitle*

        - *Query2* to search *Instance*, *Purchase*, *Format*, *Protection*, *Location* information
        - *Query3* to search *Persons* / *Roles* information related to contents.

    - In the case of **Bundle**, **2 queries** will be created:

        - *Query1* to search Bundle details, i.e. *Purchase*, *Protection, Content details*, *Instance details*, …
        - *Query2* to search *Persons* / *Roles* information related to contents in the Bundles.

5. The second interaction with the DBM now takes place (which performs the actual queries and manipulates the results).

6. Last, the response received by the QA from the DBM, is returned to the CMS.

4 DBM Servlet

- **Servlet Invocation**

  res = HTTPForwarder.doPost(<*URL_DBMServlet*>, <*npv*>)

  npv is a message (in the form of a vector) that contains the input parameters (see below)

- **Input parameters**

  - (String) **QueryID**
  - (String) **Sender** – (*at the moment only QA*)
  - (String) **TypeGet** -- ∈ (*GetIDs*, *GetDetails*, *GetTagCloud, Update*)
  - (String) **TypeCGDM** -- ∈ (*Content*, *Bundle*)
  - (String) **NumberOfQuery**
  - (String) **Query1**
  - (String) **Query2**
  - (String) **Query3** – *(only in the case of "Content")*

Explanation on some of the parameters:

- **TypeGet** indicates if we are dealing with
  - a ***Query for IDs*** *(GetIDs*, in the case of *Content* or *Bundle* determined by *TypeCGDM*),
  - a ***Query for Details*** (*GetDetails,* in the case of *Content* or *Bundle* determined by *TypeCGDM*),
  - a ***Query for TagCloud*** *(GetTagCloud)*
  - *an* ***Update*** *(Update,* as in the case of *UpdateMostViewed*; in this case only a notification update is returned)

- **NumberOfQuery** indicates the number of *Queries for Details* that have to be executed; it is defined by the QA Servlet: **3** in the case of *Content*, **2** in the case of *Bundle*
- **TypeCGDM**: depending on its value (*Content* or *Bundle*) a different CGDM structure will be created.

- **Output returned**

  - (String) **ListIDs**

  or

  - (String) **XML Response**

4.1 DBM Servlet behaviour description

1. Once invoked, the DBM Servlet extracts the **Sender** from the message. Depending on the Sender's value, it will have different behaviors – at the moment the sender is only the QA.

2. Then **TypeGet** is extracted: depending on its value (**GetTagCloud**, for searching the keywords) - **GetIDs** , for searching the IDs – **GetDetails**, for searching the details) different methods will be called

    - In the case of **TypeGet = GetTagCloud**,
        1. the query is executed;
        2. the *Aggregate* method is invoked: it will create *CGDM Java Object* from the table;
        3. then, the *CGDM Java Object* will be transformed into XML, and put into the response message returned to the QAServlet.

    - In the case of **TypeGet = GetIDs**, once the query is performed, a String is created containing the list of the IDs (*ContentIDs* or *BundleIDs*, depending on *TypeCGDM*), and returned to the QA Servlet.

    - In the case of **TypeGet = GetDetails**, depending on the value of *TypeCGDM* (*Content* or *Bundle*),
        1. the queries are executed;
        2. the *Aggregate* method is invoked: it will aggregate the tables in an optimized way and will create the *CGDM Java Object*;
        3. then, the *CGDM Java Object* will be transformed into XML, and put into the response message returned to the QAServlet.

    - In the case of **TypeGet = Update**, an update operation is executed and a notification is returned to the QA Servlet.

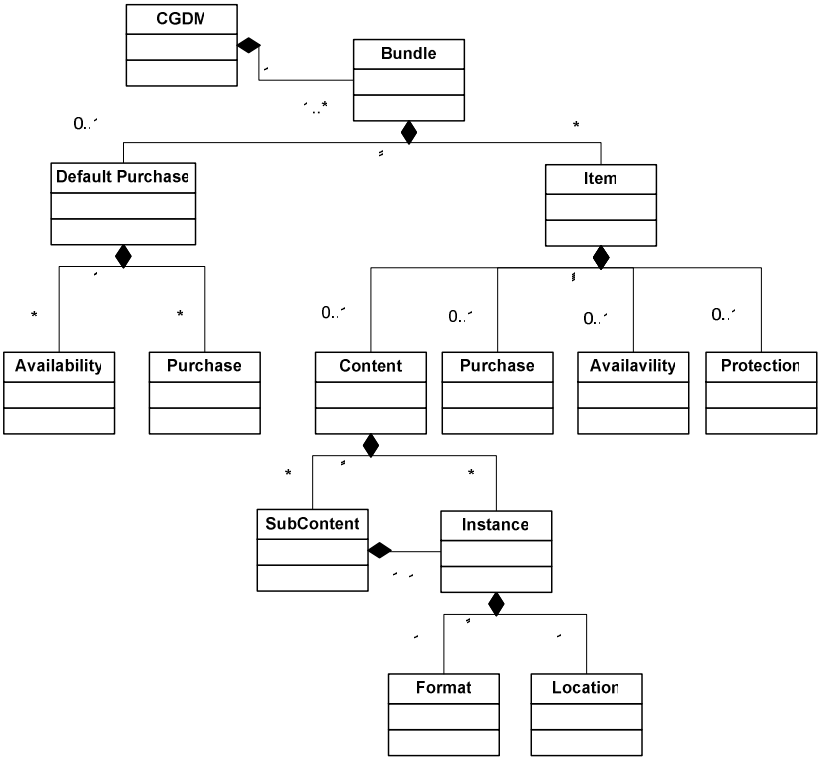
Creazione CGDM

Communication between Content Mediator Service and Content Guide Client:

Due to Dario's suggestion, we have been looking and RSS standards to see if it would fit our needs. Conclusion is that we would have to change/add so many things from the standard that it makes more sense to build our own protocol based on Content Guide Data Model described by Giuseppe.

The communication process would be:

- QA+MA receive and process a query and retrieve the results from the DB.
- It creates the following object structure (TID is implementing it)



(This is a new version from the previous one I sent last week)

- Once it is created it will be passed to the CMS and it will generate the XML for the Client to render all the information

3.2 Definizione del modello dei dati

< CGDM Data Model e relativa specifica XML>

Lo scopo di questo paragrafo è di definire il Modello dei dati del Content Guide (**SPICE Content Guide Data Model)**, che specificherà i metadati che descrivono il contenuto gestito dal Content Guide di SPICE.
Questi metadati verranno utilizzati dal Servizio di Content Guide di fornire informazioni in merito alla disponibilità dei contenuti e servizi agli utenti.
Nel presente documento il modello di dati saranno rappresentati in forma tabellare, aprendo la strada a una specifica basata su XML.

Il Data Model della Content Guide di SPICE terrà in conto, se è il caso, gli schemi dei contenuti esistenti, come ad esempio quelli dei metadati standard (MPEG-7 [1], TV- Anytime [2], CBMS ESG [3]), ampliandoli se necessario, .
Sarà necessario anche esaminare nuovi meccanismi per la ricerca attraverso il "tagging" dei contenuti per la rappresentazione della conoscenza a seguito dell' approccio "folksonomic ".

Definizioni

Regole:

- **Content provider**:
    - Per ogni contenuto si definisce il formato e l'allocazione fisica.
- **Service provider:**
    - Definisce i bundles di contents ;
    - Associa i prezzi ai bundles ;
    - Associa i prezzi ai contenuts.

Elementi:

- **Bundle**: rappresenta una collezione di contenuti aggregati da un singolo Service Provider.
- **Content**: contiene metadati che descrivono le proprietà invarianti di un pezzo di contenuto, indipendentemente da una qualsiasi particolare instanza di consegna. La descrizione dei metadata del contenuto è un informazione generale su una parte del contenuto che non cambia indipendentemente dal modo in cui il contenuto è pubblicato o trasmesso. Esso include informazioni quali il titolo del contenuto, descrizione testuale e genere. Tipicamente, il creatore del contenuto assegna la descrizione dei metadata del contenuto prima della pubblicazione.**Un "content" è una singola unità logica che l'utente può gestire.** Può essere semplice (ad esempio un video tra cui la parte audio); o composto dai principali contenuti e un certo numero di subcontent (ad esempio un video muto, come contenuto principale, con associati sottocomponenti: traccia audio in inglese + traccia audio in italiano + Traccia sottotitoli in tedesco + traccia dei sottotitoli in spagnolo,…).
- **SubContent**: Si tratta di un componente atomico del contenuto. Contiene i metadati che descrivono proprietà "invarianti" dei SubContent, indipendentemente da qualsiasi particolare istanza di consegna.
- **Format**: specifica le proprietà del formato dei contenuti (ad esempio, audio, video, immagine, applicazione). **Location**: indica l'indirizzo fisico di una istanza del contenuto.
- **Purchase**: descrive le informazioni d'acquisto (il prezzo e la moneta corrente) che possono essere associati dal Service Provider al Bundle dei content o ad una singola istanza di contenuto.
- **Protection**: descrive i diritti d'uso che possono essere associati a un 'istanza di contenuto o ad un Bundle di contenuti dal Service Provider.
- **Availability**: indica l'intervallo di tempo in cui il Service Provider mette a disposizione bundle di contenuti o una parte del contenuto.

Il rapporto tra le varie componenti del modello di dati è rappresentata nella figura 1.

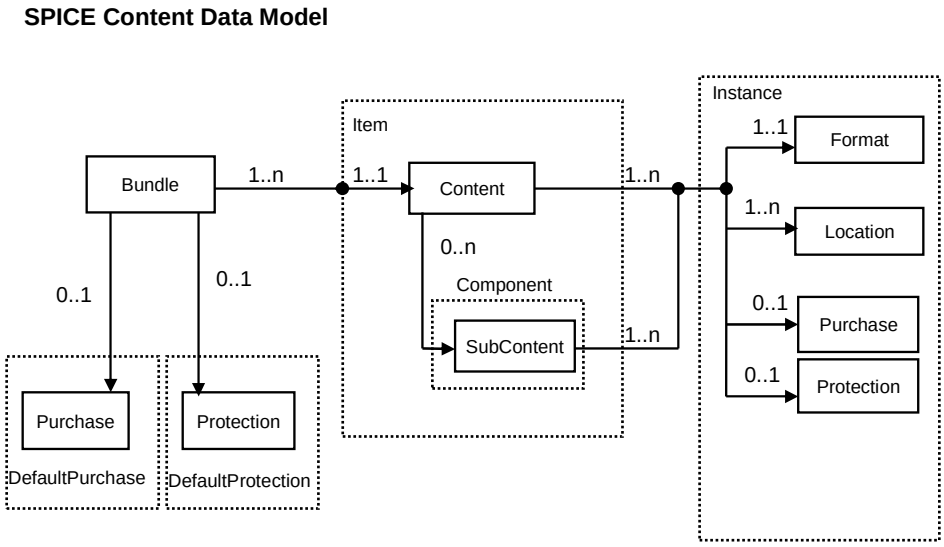


Figure 4: Content Guide Data Model Representation

<in appendice le varie tabelle>

Parlare dello schema
<in appendice lo schema>

3.2.1 Specifica e Implementazione della base dati Content Guide

Questo componente memorizza le informazioni che accompagnano il contenuto come i metadati o le informazioni giuridiche fornite dal content provider o il prezzo e le informazioni di gestione fornite dal service provider. La struttura del "metadata repository" è stato approfonditamente rivisto nel Deriverible D7.3, al fine di consentire una maggiore flessibilità nella gestione dei dati, e per tener conto di cambiamenti nel Content Guide Data Model. In primo luogo il Modello dell'entità Relazionale (**Entity Relationship (E-R) Model)** è stato progettato, individuando i soggetti e le relazioni tra di loro. Quindi il modello E-R è stato tradotto in un modello relazionale (**Relational Model** – in una rappresentazione tabellare), che a sua volta è stata implementata come un **SQL DB**. Le specifiche del DB sono disponibili in appendice C. Esse sono allineati con le specifiche del Content Guide Data Model e il corrispondente schema XML v. 9.1 (Milestone 7.1b [11]). Le tabelle relazionali di questo DB sono rappresentate nelle seguenti figure:

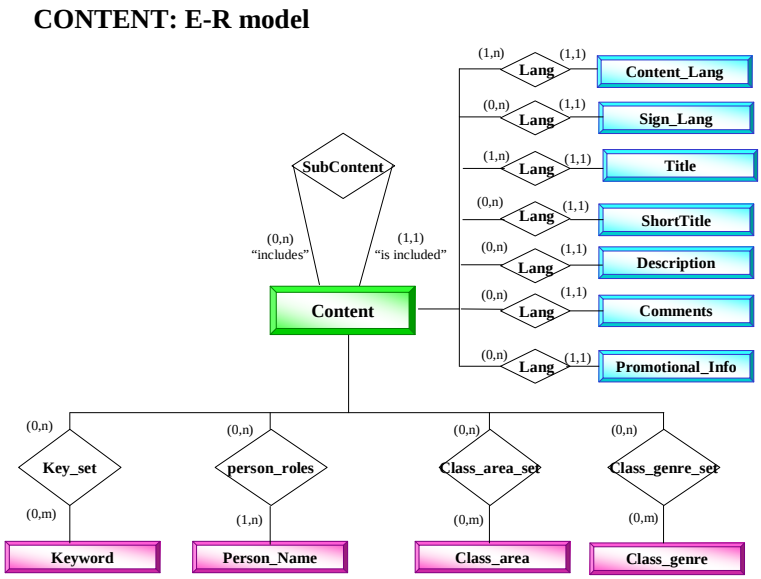


Figure 5: Content E-R model

INSTANCE: E-R model

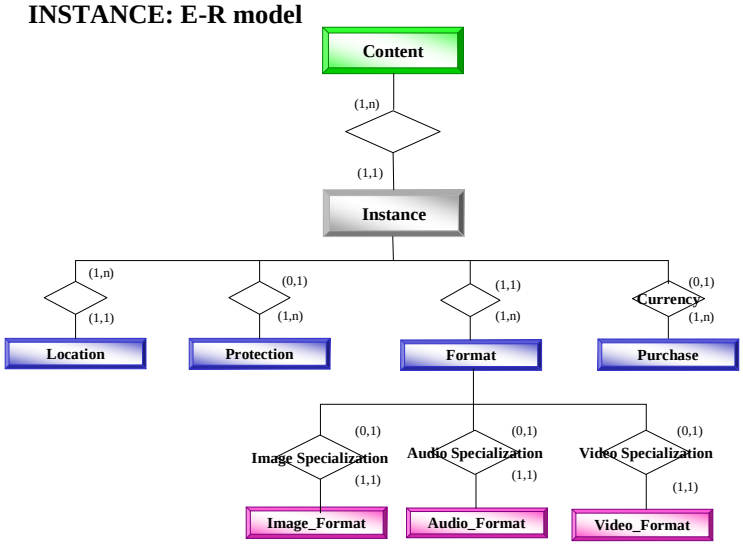


Figure 6: Instance E-R model

ITEM: E-R model

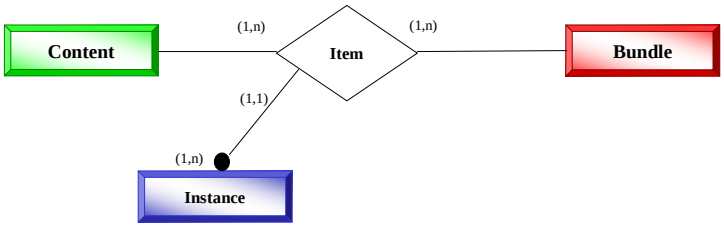


Figure 7: Item E-R model

BUNDLE: E-R model

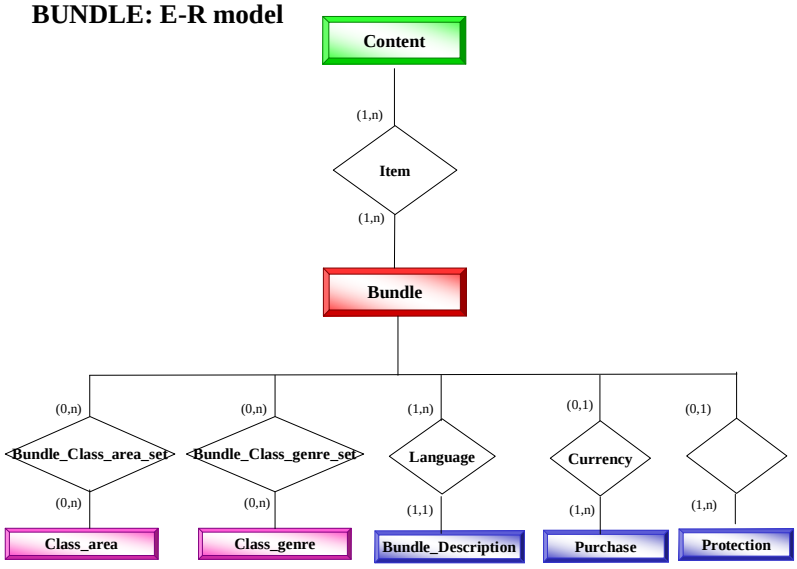


Figure 8: Bundle E-R model

Il metadata repository è stato implementato come un database MySql (v5.1). Lo SQL script per la generazione del DB è disponibile in Appendice C. La creazione di DB, attraverso l'esecuzione dello script realizzato, è stato fatto utilizzando DBTools Manager Professional.

< le tabelle in appendice>

3.2.2 Definizione dei messaggi XML

<descrizione e specifiche XML>

Una struttura flessibile del messaggio è stato definito come uno schema XML per specificare lo scambio di messaggi tra i client e CMS e tra CMS e QA.
La flessibilità dello schema consente alla stessa struttura del messaggio di poter essere applicato a tutte le richieste provenienti dagli attuali client panes, e permette di introdurre nuove e più complesse ricerche in futuro, da qualsiasi tipo di interfaccia, senza dover modificare lo schema.
Questo permette di spostare la maggior parte delle complessità di convalida dal schema XML al codice che implementa il QA, dove il Transcoder in ogni caso parsifica il messaggio XML e lo trasforma in una SQL query.
Lo schema XML introduce un tipo complesso "CGDMQueryType", definito come una sequenza di tipi CGDMQueryInstance e CGDMQueryResponse.

```
<complexType name="CGDMQueryType">
  <sequence>
    <
        element name="CGDMQueryInstance"
        type="cgdmql:CGDMQueryInstanceType"
        minOccurs="0"
        maxOccurs="unbounded"
    />
    <
        element name="CGDMQueryResponse"
        type="cgdmql:CGDMQueryResponseType"
        minOccurs="0"
        maxOccurs="unbounded"
    />
  </sequence>
</complexType>
```

Una Query Instance è caratterizzata da:

- Un **Method** (obbligatorio), che indica quale tipo di interazione è stata scelta, tra quelli disponibili dal client. Per il momento
    - *Search* – la ricerca di un valore specifico, da abbinare con appositi campi nei metadata;
    - *TagCloudReq* – corrisponde alla richiesta dell'elenco di tutti i tag e dei loro corrispondenti pesi e occorrenze, a rappresentare la "tag cloud" sul client pane;

- *TagSelection* – corrisponde alla selezione di un tag dalla tag cloud, per ottenere tutti i documenti relativi a tale etichetta ;

- *Navigation* – corrisponde alla interrogazione dell'ontologia, per ottenere l'elenco delle categorie relative a tale ontologia;

- *Selection* – corrisponde alla selezione di una categoria per ottenere tutti i documenti relativi a tale categoria.

- **Un Query Parameter** (opzionale), è composto da tre campi (opzionali):

  - **value**, indica lo specifico valore (string) che deve essere cercato;

  - **name**, indica il nome dei campi contro i quali i valori che deve essere confrontati (in generale è mappato su alcuni campi appropriati del CG Data model);

  - **type**, indica il tipo di unput. Al momento abbiamo previsto i seguenti tipi:
    - *SearchString*: indica l'input di una stringa libera ("free-text" string);
      - Se nessun parametro "name" è previsto, questa stringa verrà cercato nei confronti di tutti i campi pertinenti del DB;
      - Se il parametro "name" è fornito, la stringa verrà cercato nei confronti del campo che porta tale nome.
    - *FieldSelection*: indica che eseguiamo una selezione tra un insieme di valori predefiniti (ad esempio, di classifiche, i tipi di media,…) per limitare la ricerca;
    - *Tag*: per la ricerca del tag selezionato;
    - *OntologyCategory*: per la selezione di una categoria dall'Ontologia;
    - …
- **Logical Operators**: AND, OR, NOT, NAN, NOR

Gli operatori logici sono stati definiti per aumentare la "flessibilità" dello schema, in modo che possono essere creati messaggi di ricerca molto complessi, in una forma inequivocabile.

3.2.2.1 Query Response

Una query di risposta consisterà in:

- Documenti compatibile con lo schema CGDM - questo è ottenuta mediante l'importazione nello schema della query del namespace (spazio dei nomi) del Content Guide Data Model, cgdm="urn:spice:cgdm:2007";
- Tag Cloud info (lista dei tags e dei correspondenti pesi e occorrenze) .

Esempi

<schema in appendice – mettere qualche esempio significativo, e gli altri eventualmente in appendice>

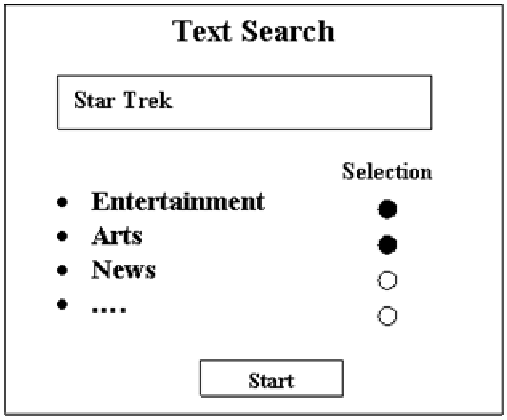
Vediamo alcuni esempi, corrispondenti agli attuali pannelli di query e di altre supplementari, per mostrare la flessibilità dello schema.

3.2.2.2 Content Guide Search

Method = "Search"

Esempio (Corrispondente al corrente Pannello di Ricerca nell'interfaccia Client)

Descrizione: vogliamo ricercare la stringa "Star Trek" e selezionare le "categories" (o la migliore classificazione) Entertainment e Art.



L'interfaccia non fornisce alcuna indicazione sui campi specifici contro i quali la stringa deve essere confrontata: in questo caso il Transcoder sa che dovrà ricercare la stringa "Star Trek" nei confronti di campi pertinenti nel DB (in generale tutti coloro che contengono stringhe, ad esempio "Titolo", "Artista", "Commento", "Descrizione:,….).
La selezione dei valori di classificazione (vale a dire Arte, Entertainment), corrisponderà a abbinamento con i campi della Classificazione (abbiamo convenuto che solo Classificazione Area e Genre verranno utilizzati).

```
<?xml version="1.0">
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
<CGDMQueryInstance method="SEARCH">
        <AND>
                <CGDMQueryParameter type="SearchString"   value="Star Trek" />
                <OR>
                        <CGDMQueryParameter type="FieldSelection" name="Classification" value="Entertainment" />
                        <CGDMQueryParameter type="FieldSelection" name="Classification" value="Art" />
                </OR>
        </AND>
</CGDMQueryInstance>
</CGDMQuery>
```

Corresponding response message

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
        <CGDMQueryResponse>
                <CGDM>
                        <!--XML Document compliant to CGDM schema -->
                </CGDM>
        </CGDMQueryResponse>
</CGDMQuery>
```

3.2.2.3 Esempio (Richiesta informazioni Bundle)

Method “Search” - Type “SearchBundles”

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
      <CGDMQueryInstance method="SEARCH">
      <AND>
            <CGDMQueryParameter type="SearchBundles" name="Content_ID"
value="<selected Content_ID value>"/>
      </AND>
      </CGDMQueryInstance>
</CGDMQuery>
```

Dove

- Il metodo è SEARCH ;
- Il Parametro type della query è "SearchBundles" ;
- Il Parametro name della query è "Content_ID", così abbiamo recuperato tutti i Bundle che contengono un certo contenuto ;
- Il Parametro value della query è l’URI corrispondente al Content_ID del messaggio di risposta corrispondente.

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
      <CGDMQueryResponse>
            <CGDM>
                  <!--XML Document compliant to CGDM schema -->
            </CGDM>
      </CGDMQueryResponse>
</CGDMQuery>
```

3.2.2.4 Esempio (Ricerca dei Content_IDs per oggetti raccomandati)

Descrizione: Dato un elenco di Content_IDs restituiti al CMS (per esempio dal sistema di raccomandazione), vogliamo ottenere tutti i dettagli relativi a tali contenuti.

Method “Search” - Type “Recommender”

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
      <CGDMQueryInstance method="SEARCH">
      <AND>
            <CGDMQueryParameter type="Recommender" name="Content_ID"
value="Val1"/>
            <CGDMQueryParameter type="Recommender" name="Content_ID"
value="ValN"/>
      </AND>
      </CGDMQueryInstance>
</CGDMQuery>

<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
      <CGDMQueryInstance method="SEARCH">
      <AND>
            <CGDMQueryParameter type="Recommender" name="Content_ID"
value="Val1"/>
            <CGDMQueryParameter type="Recommender" name="Content_ID"
value="ValN"/>
      </AND>
      </CGDMQueryInstance>
</CGDMQuery>
```

Messaggio di risposta corrispondente

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
        <CGDMQueryResponse>
                <CGDM>
                        <!--XML Document compliant to CGDM schema -->
                </CGDM>
        </CGDMQueryResponse>
</CGDMQuery>
```

3.2.2.2 Tags

Methods *€* (“TagCloudReq”, “TagSelection”)

3.2.2.3 Query Example (Tag Cloud)

Descrizione: richiede la lista deiTags con I relative pesi.

```
<?xml version="1.0"?>
    <CGDMQuery xmlns="urn:spice:cgdmql:2007">
            <CGDMQueryInstance method=" TagCloudReq ">
    </CGDMQueryInstance>
</CGDMQuery>
```

Messaggio di risposta corrispondente

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
    <CGDMQueryResponse>
            <TagCloud>
                    <TagItem value="Tag" weight="23"></TagItem>
                    <TagItem value="Tag2" weight="34"></TagItem>
                    <TagItem value="TagN" weight="66"></TagItem>
            </TagCloud>
    </CGDMQueryResponse>
</CGDMQuery>
```

3.2.3 Esempio (Selezione del Tag dal Cloud)

Descrizione: di uno specifico Tag

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
    <CGDMQueryInstance method="TagSelection">
            <AND>
                    <CGDMQueryParameter type="Tag" value="TagN" />
            </AND>
    </CGDMQueryInstance>
</CGDMQuery>
```

Messaggio di risposta corrispondente

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
    <CGDMQueryResponse>
            <CGDM>
                    <!--XML Document compliant to CGDM schema -->
            </CGDM>
    </CGDMQueryResponse>
</CGDMQuery>
```

3.2.4 Most Viewed

Method = "MostViewed"

3.2.5 Esempio (Ricerca Most Viewed – corresponding to Most Viewed from My CG Pane)

Descrizione: Dato un elenco di Content_IDs restituiti al CMS (per esempio dal sistema di raccomandazione), vogliamo ottenere tutti i dettagli relativi a tali contenuti

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
      <CGDMQueryInstance method="MostViewed">
      </CGDMQueryInstance>
</CGDMQuery>
```

Messaggio di risposta corrispondente

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
        <CGDMQueryResponse>
                <CGDM>
                        <!--XML Document compliant to CGDM schema -->
                </CGDM>
        </CGDMQueryResponse>
</CGDMQuery>
```

3.2.6 Content Selection

Methods "ContentSelection"

3.2.7 Esempio (Corrispondente al pannello Play nell'interfazzia Client)

Descrizione: Richiesta di "play" per il content selezionato.

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
    <CGDMQueryInstance method="ContentSelection" id="URI">
            <AND>
            <CGDMQueryParameter type="Price" value="XXX"/>
            </AND>
    </CGDMQueryInstance>
</CGDMQuery>
```

Messaggio di risposta corrispondente inviato al Client

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
    <CGDMQueryResponse>
            <CBResponse>
                    <sessionID>12</sessionID>
                    <payment_method id="pm2">
                            <rank> 2 </rank>
                            <description> internet_operator_bill </description>
                            <billingMode>prepaid</billingMode>
```

```
                    </payment_method>
                </CBResponse>
            </CGDMQueryResponse>
        </CGDMQuery>
```

3.2.8 Play

Method “Play”

3.2.9 Esempio (Corrispondente al pagamento completato nel pannello di conferma)

Descrizione: Pagamento per il content selezionato e play

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
    <CGDMQueryInstance method="Play" id=”URI”>
    <AND>
            <CGDMQueryParameter type="PaymentMethod" name="Method1"/>
            <CGDMQueryParameter type=”SessionContext” value=”12”/>
    </AND>
    </CGDMQueryInstance>
</CGDMQuery>
```

Messaggio di risposta corrispondente inviato al Client

3.2.10 Feedback

Method “Feedback”

3.2.11 Esemple (Corrispondente alla valutazione del Rating)

Descrizione: restituisce informazioni sul rating associate al contenuto scaricato.

```
<?xml version="1.0"?>
<CGDMQuery xmlns="urn:spice:cgdmql:2007">
    <CGDMQueryInstance method="Feedback" id=”URI”>
            <AND>
            <CGDMQueryParameter type="Rating" value="1"/>
            </AND>
    </CGDMQueryInstance>
</CGDMQuery>
```

Messaggio di risposta corrispondente inviato al Client

3.2.12 Tabella delle Corrispondenze

La seguente tabella indica la possibile corrispondenza tra i campi dei parametri di ricerca: Tipo / Nome / Valore, indicando i possibili valori che possono assumere, nel pieno rispetto del Content Guide DataModel.

Type	Name	Value
SearchString	-	“string”
	Artist	“string”
	Title	“string”
	ShortTitle	“string”
	Description	“string”
	Author	“string”
	CharacterName	“string”
	Director	“string”
	Geotagging	?
SelectField	Classification_Area	Entertainment, Art, …
	MediaType	Video,Audio, Image, Text
	Language	EN, IT,
Tag	-	“string”
OntologyCategory	“name_of_the_category”	-
PaymentMethod	“name_of_the_payment_method”	“price of the payment method”
Price	-	“string”
SessionContext	-	“string”
Rating	-	“string”
SearchBundles	Content_ID	“string” (e.g Content_ID value)
Recommender	Content_ID	“string” (e.g Content_ID value)

4 Realizzazione del servizio di Content Guide con Lucene

4.1 Il motore di ricerca Lucene

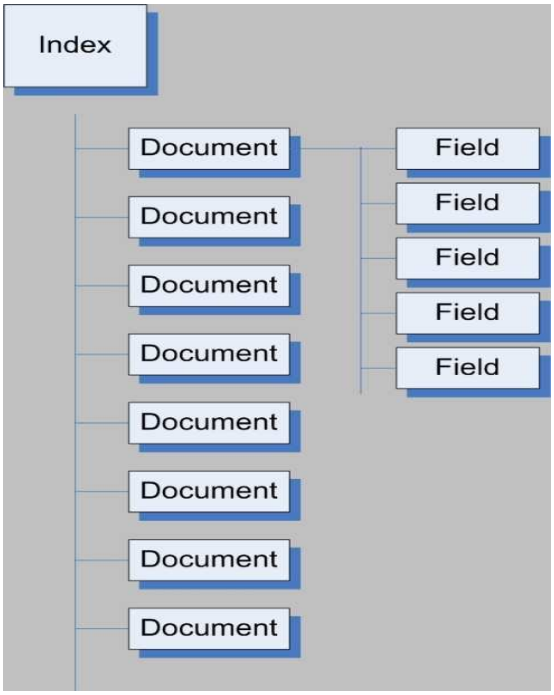
Lucene è la famosa libreria open source che permette di avere in Java un motore di ricerca per diverse tipologie di file. E' un progetto open source della Apache Software Foundation scritto da Doug Cutting. Lucene è un' API estremamente flessibile che ci permette di inserire nelle nostre applicazioni le funzionalità di motore di ricerca. Sebbene concepito per realizzare applicazioni che necessitano di funzionalità di indicizzazione e ricerca full text , Lucene è molto nota ed usata per la realizzazione di motori di ricerca sia sul World Wide Web che sulle intranet private. Questo ha portato all'affermazione di una percezione del prodotto come di un motore di ricerca dotato di web spider e parser HTML. In realtà, l'utilizzo di lucene richiede che tali moduli siano forniti esternamente. I dati gestiti da Lucene sono rappresentati come *documenti* (Document) dotati di *campi* (Fields) testuali. Questa genericità consente di realizzare con le API di Lucene prodotti che sono indipendenti dal formato dei file: possono essere indicizzati con Lucene testi in PDF, HTML, Microsoft Word, così come ogni altro tipo di file dal quale è possibile estrarre informazioni. Per maggiori informazioni su download, licenze e quant'altro si rimanda al sito ufficiale http://lucene.apache.org/java/docs/index.html.

4.1.2 Una panoramica su Lucene

Lucene viene chiamata, con un termine che può essere fuorviante: "Indice"; prendendo il confronto con i database relazionali possiamo dire che ogni indice contiene una sola tabella le cui righe altro non sono che i documenti da noi inseriti. L'indice è normalmente contenuto in una cartella del filesystem, ma esistono altre possibilità, quali la creazione di un indice in RAM, utilizzato soprattutto per aumentare le prestazioni (con particolare attenzione al limite imposto dalle dimensioni della memoria). All'interno dell'indice vengono inseriti documenti (istanze di org.apache.lucene.document.Document), a loro volta divisi in campi o colonne (Fields); come nei database esistono vari tipi di campi, anche se in questo caso il tipo di dati è lo stesso per ogni colonna (o quasi, in realtà esistono alcune piccole differenze).
I campi testuali veri e propri (come il titolo ed il corpo di un documento) verranno indicizzati , mentre altri campi, ad esempio l'identificatore del documento all'interno di un database, saranno solo salvati, per essere utilizzati come riferimento.
In pratica non sarà possibile eseguire ricerche in questi campi, ma solo accedere al loro valore, nei documenti ritornati da una query eseguita su un altro campo. Lo scenario di utilizzo prevede infatti nella quasi totalità dei casi l'affiancamento della base di dati full-text ad una relazionale. Così facendo utilizzeremo Lucene per fornire un sistema di ricerca (e per presentarne i risultati), mentre quando l'utente desiderà visualizzare un documento (ad esempio cliccando sul titolo) accederemo ad una pagina popolata dal database relazionale; sfrutteremo Lucene per l'efficienza delle ricerche ed il DBMS per la maggiore velocità nella selezione di un singolo elemento (aiutato in questo magari da un opportuno meccanismo di caching, attivo e passivo).

4.1.2.1 Caratteristiche

Lucene presenta una vasta collezione di interfacce che definiscono tutti i classici passi della ricerca. Questa libreria quindi non è un vero e proprio motore di ricerca che noi richiamiamo dicendogli "Mi cerchi questa cosa?" e lui fa tutto in automatico.
Proprio per questo motivo da una parte non viene subito capito dagli sviluppatori. Bisogna però rendersi conto della flessibilità delle API che vengono messe a disposizione da Lucene. Infatti una volta capito il modo in cui possiamo adattare i dati che vogliamo ricercare al pattern di Lucene, il suo utilizzo diventa molto semplice.
Ci sono degli oggetti con i quali bisogna prendere confidenza, perché saranno gli attori principali nell'utilizzo di questa libreria. L'immagine seguente riassume queste entità e le loro relazioni.



Strutturazione indice di Lucene.

Come potete vedere la gerarchia delle nostre informazione parte dall'"INDEX", ovvero dall'indice. In questo sono presenti diversi "DOCUMENT", che rappresentano i vari documenti che sono stati indicizzati. Per ogni documento avremo diversi "FIELD", ovvero una coppia di nome/valore che identifica un'informazione sul nostro documento. Nel momento in cui noi dobbiamo creare un indice dobbiamo utilizzare la classe IndexWriter di Lucene, che ci permette appunto di aggiungere tutti i Document al nostro indice. Il Document stesso deve essere creato da noi, inserendo al suo interno i diversi Field che lo descrivono.
Quando dobbiamo effettuare una ricerca ci dobbiamo basare sul QueryParser, che è la classe

generatrice delle nostre Query. Come risultato di una query avremo un vettore di Document, che soddisfano i requisiti della nostra ricerca. Lucene è una libreria open source per creare un potente motore di ricerca per siti e applicazioni. Vediamone aspetti fondamentali e un confronto con le caratteristiche dei più comuni DBMS. Le funzioni di trattamento di dati testuali sono una parte importante in un numero di applicazioni sempre più alto; dal CMS al più semplice software gestionale è infatti necessario permettere all'utente di interagire con delle risorse testuali. Da alcuni anni esiste un componente open-source che permette di realizzare soluzioni estremamente avanzate, per l'appunto "Apache Lucene".

4.1.3 DBMS vs Full-Text

Possiamo evidenziare con grande efficacia le possibilità offerte da una soluzione full-text partendo da un confronto con le soluzioni presenti nei sistemi maggiormente utilizzati dagli sviluppatori, le basi di dati relazionali.
Di solito siamo abituati a lavorare con i DBMS basati sul modello relazionale che, come è ben noto, prevede una rappresentazione dei dati in tabelle, le cui righe rappresentano gli elementi inseriti e le cui colonne sono dei campi cui viene assegnato un valore; all'interno di queste tabelle possiamo effettuare operazioni di selezione ed ordinamento, basandoci sui valori di alcune colonne, che contengono valori numerici (o pseudonumerici).
Pensiamo, ad esempio, ad un sito Web di carattere giornalistico: il sistema di presentazione molto spesso si avvarrà di una base di dati, partendo dalla quale selezionerà gli articoli da presentare ai visitatori sulla base dell'identificativo con cui sono salvati nel database, per selezionare gli ultimi elementi pubblicati, oppure sulla base di colonne che specificano la categoria della notizia (per fornire più pagine, ad esempio una dedicata all'economia ed una allo sport).
Come è possibile notare dalle considerazioni appena fatte il database relazionale serve per lo più a garantire la persistenza dei dati testuali, mentre poco può fare per quanto riguarda la loro elaborazione.
Volendo eseguire invece una ricerca all'interno dei testi, per trovare quelli che contengono un certo termine, possiamo utilizzare l'operatore SQL 'LIKE'; questa soluzione presenta però notevoli limiti, in termini di efficienza e di prestazioni.
Per fornire all'utente un sistema più avanzato di ricerca quasi tutti i DBMS forniscono dei moduli o delle estensioni full-text: attivando queste funzionalità è possibile accedere a servizi avanzati di ricerca testuale. Ad esempio in MySql, una volta definita l'indicizzazione full-text è possibile eseguire interrogazioni mediante le parole chiave MATCH() e AGAINST().
Esistono però altre soluzioni che consentono di estendere le proprie applicazioni per fornire servizi avanzati di analisi testuale: tra questi prodotti software Apache Lucene è uno dei più apprezzati e potenti.
Ma quali sono i motivi concreti per cui può essere opportuno appoggiarsi a prodotti specifici per il trattamento di testi?
Le ragioni sono molteplici, ma preferisco soffermarmi sulle due che ritengo più importanti. In primo luogo molto spesso ci si può trovare a confrontarsi con banche dati testuali che contengono fino a diverse decine di migliaia di documenti: in questa situazione le prestazioni ottenute tramite un motore ad hoc sono nettamente superiori rispetto a quelle raggiungibili con le espansioni full-text dei DBMS commerciali. Come esperienza personale posso dire di aver visto situazioni in cui uno dei più diffusi DBMS enterprise ha mostrato evidenti limiti a livello di prestazioni, soprattutto

se la base documentale si caratterizza per un numero molto elevato di ricerche full text con incrocio di dati su tabelle diverse.

L'altro motivo, come è prevedibile, risiede nelle possibilità offerte da una soluzione full-text pura come Lucene: ricerche complesse con operatori booleani, ricerche all'interno di una specifica frase (i termini della query devono apparire vicini o ad una distanza massima tra loro), ricerche con wildcard e così via.

Un altra delle feature distintive dei sistemi IR (Information Retrival) in confronto con i DBMS (Data Base Management System) è che i primi offrono miglior compressione per le posting list, che risulta in miglior performance di I/O e quindi maggior velocità nella valutazione delle query.

4.2 Architettura di Sistema

Le modifiche apportate riguardano pochi elementi dell'architettura del progetto originario sviluppato in Telecom.
Una prima modifica è stata imposta data la notevole dimensione del DBLP.
Abbiamo pensato di creare un parser ad hoc per la costruzione dell'albero degli element, che suddivide il file XML della bibliografia in un insieme di file di dimensione più piccola.
Il vantaggio che abbiamo ottenuto è quello di sfruttare a pieno la RAM evitando che la dimensione della struttura dati intermedia ecceda lo spazio consentito della memoria principale.
Dopo la fase di parsing, in cui viene creato un albero di element, invece di creare un CGDMElement Tree come in precedenza, si creerà un DocumentTree. Questa struttura intermedia di supporto conterrà per ogni nodo un oggetto chiamato DocElement che modellerà le istanze contenute nel DBLP e tutte le informazioni che le caratterizzano: ad esempio: il Nome dell'Autore, il titolo del libro e l'anno di pubblicazione, ecc.
Il vettore di DocElement è stato pensato come un albero piatto, cioè ad un solo livello di profondità. Questa scelta ha portato a due vantaggi:

- aderire con semplicità alla successiva fase di creazione dell'Index di Lucene,
- attraverso l'utilizzo di un Hashmap, abbiamo eliminato i ritardi introdotti dagli accessi in memoria per il reperimento di indici degli autori e degli editori;

Il DB è stato leggermente modificato per rispettare quelle che sono le regole principali per la modellazione dei dati, nel nostro caso il DBLP. E per la modellazione delle relazioni M:N.

4.3 Specifica e Implementazione delle componenti SW

Utilizzo il metodo parserXML1 della classe Parse che prende in ingresso una stringa e ritorna il puntatore all'albero di Element. Creo la classe Transcode, duale alla classe QueryTranscoder, che trasforma l'albero di Element in un albero di DocElement.
L'oggetto DocElement contiene al suo interno:

- un vettore per ogni tipo di informazione (es. un vettore per "author", ecc) in modo da supportare la stuttura del file .dtd;
- una Map che consente di manipolare con agilità tutti i vettori.

Ottenuto il DocumentTree si passa alla creazione dell'index di Lucene attraverso la classe CreateIndexLucene, dove per ogni oggetto Document dell'albero precedentemente creato si instanzierà un nuovo Document dell'index i cui Fields conterranno le informazioni relative ad ogni vettore dell'oggetto DocElement.

La Classe Storage è stata creata per riempire il DB.
Essa a partire dal vettore di DocElement, per ogniuno DocElement crea un vettore di stringhe, che contengono le insert da eseguire sul DB per riempirlo.

5 Lucene vs Mysql

Al fine di confrontare le prestazioni del motore di ricerca Lucene con la piattaforma Mysql sono stati effettuati dei test.
Tali test sono stati tradotti nei rispettivi linguaggi di query (vedi in appendice Tabella della delle query) ed eseguiti su “roquefort.di.unipi.it”, una della macchine messe a disposizione dal Dipartimento di Informatica di Pisa. Le caratteristiche tecniche di “Roquefort” sono: biprocessore Intel(R) Pentium® 4 CPU di 2.60 GHz, con RAM da 1,5 GB. Sistema Operativo Linux. La versione di java utilizzata è xxxx. La versione di mysql utilizzata è yyyy.

Per i nostri test abbiamo scelto di utilizzare, come insieme di dati consistenti, i dati bibliografici delle pubblicazioni di Computer Science tratti dal sito “DBLP Computer Science Bibliography” che li fornisce anche in formato XML. Il file utilizzato è stato scaricato il 02/09/2008 (il sito aggiorna costantemente la bibliografia), ha dimensione di circa 470MB e contiene 1104700 istanze.

Nei prossimi paragrafi analizzeremo la fase di creazione del repository e le prestazioni raggiunte durante le interrogazioni.

5.1 Creazione Repository

Nella tabella di seguito riportiamo le prestazioni delle piattaforme Lucene e Mysql per la creazione del Repository. Da notare che sono stati effettuati test sulla creazione del DB con e senza interfaccia java, in quest'ultimo caso colloquiando direttamente col gestore del DB. Le misurazioni sono frutto di ripetuti esperimenti e rappresentano l'andamento medio dei rispettivi comportamenti. Tali test sono stati effettuati sulla macchina "roquefort.di.unipi.it" messa a disposizione dal dipartimento di Informatica di Pisa come citato nel paragrafo precedente.

	Tempo Creazione in sec.	Dimensione in GB	Working space medio in MB	**Working space massimo in MB**
Lucene	2020,234 (circa 0:33 ore)	0,488	1416393	1537104
MYSQL (con interfaccia java)	9955,282 (circa 2:45 ore)	1,334	1419635	1536952
MYSQL (senza interfaccia java)	9322, 181 (circa 2:35 ore)	1,334	1419584	1536892

Tabella 5.1.1
Creazione Repository

Come si evince dalla tabella 5.1, i tempi per la creazione del Repository differiscono notevolmente nelle diverse piattaforme.
La costruzione dell'indice di Lucene è cinque volte più veloce del tempo impiegato da Mysql per la creazione del DataBase.
Ripetuti esperimenti hanno dimostrato che tale differenza resta pressocchè invariata anche in assenza dell'overhead introdotto dall'interfaccia java che è di circa 10 minuti su un DB di 1,334 GB.

Anche la dimensione dei rispettivi repository differiscono; ciò è dovuto alla diversità degli approcci utilizzati per la costruzione delle proprie strutture dati.
Lucene, essendo utilizzato come motore di ricerca, sfrutta una struttura piatta che riduce fino a tre volte, rispetto a quella del DB, la dimesione del repository. Si può notare che la sua dimensione è lineare con quella del file originale (che è di 470 MB) in quanto differiscono di circa 18 MB.
La dimensione del DB, al contrario dell'indice di Lucene, è triplicata rispetto ai dati originali.

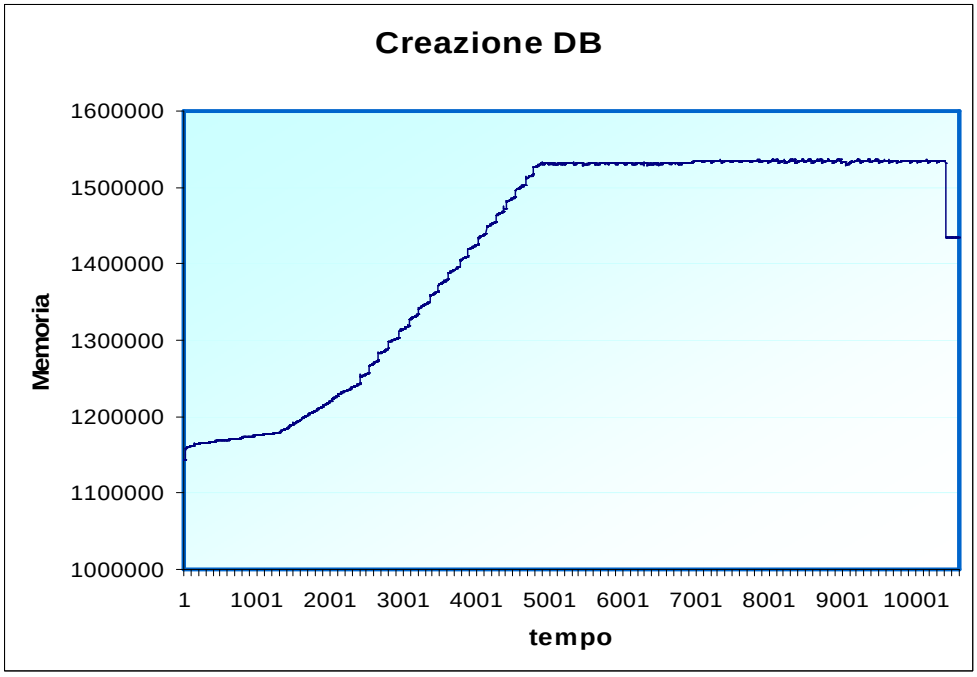


Tabella 5.1.2 Creazione Database.

Abbiamo analizzato il comportamento spazio-temporale per la costruzione del Repository e come si nota dal grafico l'andamento di MYSQL è quello atteso. La creazione del DataBase raggiunge il picco di utilizzo della RAM dopo circa 5000 sec.

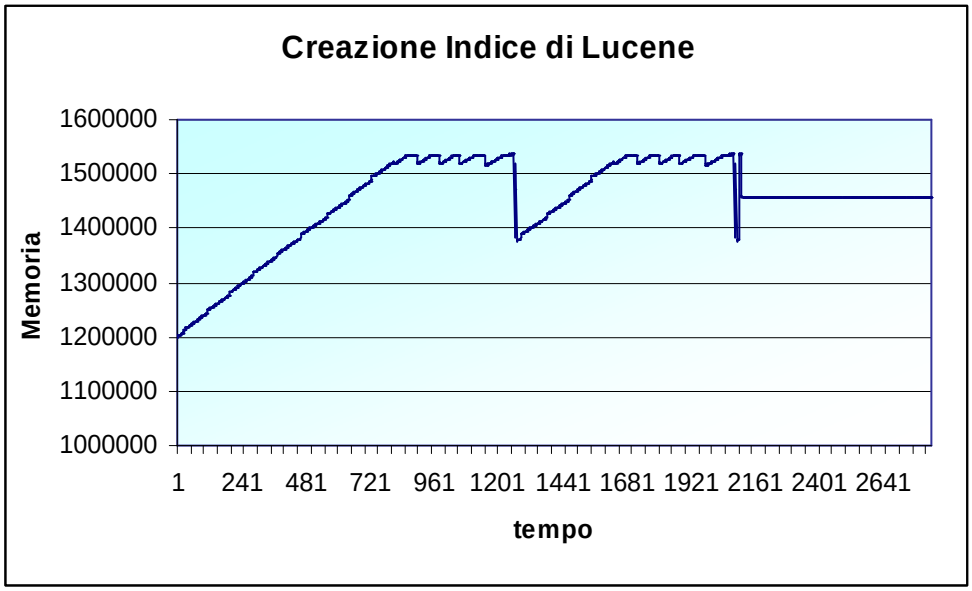


Tabella 5.1.3 Creazione Indice di Lucene.

Al contrario il comportamento di Lucene è apparentemente sorprendente per via degli avvallamenti che periodicamente si verificano.
Questo fenomeno è dovuto all'approccio utilizzato dal motore di ricerca per la creazione del suo indice, detta tecnica di "doubling".
Durante la fase si costruzione dell'indice, ogni qual volta che si aggiunge un nuovo Document, invece di scriverlo in memoria secondaria lo bufferizziamo. Raggiunto un certo numero di Document (il valore limite dipende dalle caratteristiche della macchina e possono essere impostate a linea di comando), Lucene crea un file progressivo su Disco, chiamato "segmento", che li contenga e a quel punto libera memoria RAM.
Gli avvallamenti che si notano nel Grafico rispecchiano in pieno questa caratteristica.
Inoltre la possibilità di raggruppare i document in segmenti scaturisce dalla loro indipendenza rispetto ai dati di origine; da qui il vantaggio che si trae deriva dalla velocità di aggregazione dei dati, quindi dall'eliminazione del ritardo introdotto dall'accesso in Memoria secondaria sostituito dalla loro bufferizzazione in RAM.
Infine, si applica all'insieme dei segmenti così ottenuti la fase di concatenazione da cui deriva l'intero indice e la fase di ottimizzazione in cui i dati vengono compressi per ottimizzare lo spazio su disco.

5.2 Prestazioni Repository

Di seguito vengono riportate la tabella delle prestazioni comparate e la tabella delle query eseguite, in formato XPath.

	Query	Totale Risultati	Tempi di risposta	
			MySql	Lucene
query frequenti >= 10 000	Q1	412.224	11,77 sec	0,29 sec
	Q2	29.641	5,81 sec	0,24 sec
	Q4	10.394	7,03 sec	0,22 sec
	Q5	23.523	101,13 sec (~1 minuto e 41,13 sec)	0,33 sec
Query selettive 100-1000	Q3	64	3,91 sec	0,60 sec
	Q6	593	84,84 sec (~1 minuto e 24,84 sec)	0,47 sec
	Q7	237	28,78 sec	1,35 sec
	Q8	301	29,31 sec	0,69 sec

Tabella 5.2.1 Prestazioni Repository.

query	X-Path
Q1	//article
Q2	//*/year=”1995”
Q3	//*/booktitle=”Modern*”
Q4	//*/title=”Theory”
Q5	//*/author=kim*”
Q6	//*/author=bertin*
Q7	//*/year=2007 and author=Yung
Q8	//improceedings/author=”Yung”

Tabella 5.2.2 delle query in XPath.

Riportiamo solo alcuni dei test, quelli più significativi, che sono stati effettuati per la misurazione delle prestazioni delle rispettive piattaforme. Tali test sono stati ripetuti più volte al fine di individuarne il comportamento medio.
Dalle misurazioni rilevate abbiamo notato che per query frequenti (sopra i 10000 risultati ottenuti) il tempo di risposta di Lucene è nettamente inferiore ripetto a quello impiegato da Mysql che è da 24 a 300 volte più breve del suo concorrente. Le rilevazioni confermano che i forti ritardi di Mysql

crescono all'aumentare del numero di tabelle da attraversare, introdotte dove necessario per modellare le relazioni M:N. Al contrario Lucene, avendo una organizzazione della struttura dati pressocchè piatta, esegue una scansione lineare delle liste invertite, da cui le ottime prestazioni.
Anche nel caso di query selettive (da 100 a 1000 risultati ottenuti) Lucene è più veloce di Mysql ; i suoi tempi di risposta sono da 6 a 180 volte più brevi.

5.2.1 Le Query

Analizziamo ora le query che abbiamo utilizzato nei test per misurare le prestazioni delle piattaforme che sono riportate per comodità in X-Path.
In appendice è riportata la tabella delle query eseguite tradotta nei rispettivi linguaggi di programmazione.
Va notato che tutte le query hanno richiesto alcuni aggiustamenti sintattici per rispettare la semantica delle interrogazioni.

Query esatta
Questa semplice query è principalmente per stabilire una unità di performance come metro di giudizio sul quale valutare tutte le query successive.

Q1: //article
Seleziona tutti i contenuti che sono classificati esattamente come "article" .

Query per Attributo
Questa query è stata scelta per testare la velocità di MySql nell'attraversare una sola tabella.

Q2: //*/year="1995"
Ritorna tutti i contenuti che sono dell'anno 1995.

Query Approssimata 1
Oltre alla ricerche esatte , abbiamo scelto di testare le prestazioni anche su ricerche approssimate.
In questo caso la chiave inserita potrebbe essere parte prefissa di una parola all'interno della stringa cercata.

Q3: //*/booktitle="Modern*"
Ricerca tutti i contenuti che hanno "Modern*" come prefisso di almeno una delle parole che compongono il titolo della collana.

Query Approssimata 2
Ricerca i contenuti che hanno esattamente la parola "Theory" che occorre nella stringa

Q4: //*/title="Theory"
Ritorna i contenuti che hanno esattamente la parola "Theory" che occorre nel titolo.

Query Approssimata 3
Questa query è stata scelta per constatare il ritardo introdotto sia nell'attraversare tre tabelle e sia per la ricerca approssimata.

Q5://*/author=kim*"
Ricerca tutti gli autori che hanno "kim" come nome o parte del nome.

Query Approssimata 4
In questa questa query è stato scelto un nome italiano per osservare i tempi di risposta di MySql nel caso di pochi dati ottenibili.

Q6://*/author=bertin*
Ricerca tutti gli autori che hanno "bertin" come nome o parte del cognome.

Query complessa 1
La query ha l'obiettivo di eseguire una ricerca più complessa che attraversi 3 tabelle e testarne i tempi di risposta.

Q7://*/year=2007 and author=Yung
Ricerca tutti i contenuti che hanno come autore "Yung" e sono dell'anno 2007.

Query complessa 2
La query ha l'obiettivo di eseguire una ricerca più complessa che attraversi 5 tabelle per testarne i tempi di risposta.

Q8://improceedings/author="Yung"
Ricerca tutti i contenuti che sono etichettati come "improceedings" e hanno come autore "Yung".

5.3 Conclusioni

La maggior parte dei sistemi IR e dei motori di ricerca web usa liste invertite, che sono provate essere molto efficienti per rispondere ad interrogazioni. Comunque negli ultimi anni la varietà di servizi che questi sistemi offrono (o dovrebbero offrire) si sta ampliando. Per esempio dovrebbero essere in grado di gestire dati strutturati (Google Base), documenti strutturati o dati semi-strutturati (XML), annotazioni, tag e tipi di dato multimediali. Inoltre vengono eseguiti una grande varietà di nuovi compiti, molto differenti dai classici task di valutazione delle query, fra cui algoritmi di data mining, machine learning, collaborative reccomendation, ecc...

Per queste ragioni, gli indici dei motori di ricerca dovrebbero essere facilmente estendibili e in grado di accogliere vari tipi di dati e metadati. Le tipiche amenità che un DBMS offre (linguaggi di query dichiarativi, ottimizzatori di query) sono certamente utili quando si ha a che fare con svariati tipi di dato.

Mettiamo a confronto i DBMS contro i file invertiti e traiamo le nostre conclusioni.

Per quanto riguarda i DBMS, un vantaggio è l'estendibilità dell'ambito dell'indice. Poichè l'ambito di servizio dei sistemi IR e dei motori di ricerca web si sta costantemente ampliando, è importante che essi siano basati su indici facilmente estendibili.
Nel caso di un indice DMBS, l'estensione dello schema dell'indice con ulteriori colonnee relazione è abbastanza immediata.
Per esempio, un indice potrebbe essere espanso con varie informazioni, come utenti, date, tag, metadati, con lo scopo di supportare query più sofisticate e modelli di ricerca.
Al contrario, i file invertiti sono stati specificatamente creati per fornire accesso molto veloce basato su termini, e allargare lo spettro del loro contenuto e delle loro funzionalità è un task laborioso e nel migliore dei casi genera soluzioni ad hoc.

Un beneficio di usare un DBMS riguarda il processo di costruzione dell'indice. Poichè il layer fisico è gestito dal DBMS, non c'è bisogno di creare e riunire file parziali per poter ricreare l'indice di una grossa mole di dati. Ricordate che per moli di dati abbastanza grosse, il file invertito non entra in memoria principale e deve quindi essere usata una qualche tecnica di ordinamento parziale.

Ancora, un DBMS può facilitare la manutenzione dell'indice. In generale, molte pagine web appaiono e scompaiono velocemente. Come conseguenza, la manutenzione dell'indice può essere un fattore importante per l'adozione di un indice specifico, poichè dover ricostruire completamente l'indice troppo spesso può essere molto costoso.
Specificatamente, cancellare le entry che riguardano un particolare documento è un'operazione molto costosa in un file invertito, il suo costo è O(n), dove n è la dimensione in parole della collection. In lavori recenti questo costo è stato ridimensionato, ma rimane un'operazione costosa.

Infine, i sistemi IR stanno diventando più complessi per poter avvantaggiarsi dell'attuale hardware multicore o multiprocessore. D'altra parte i sistemi DBMS sfruttano fino in fondo da anni tali caratteristiche.

Per quanto riguarda le liste invertite, vediamo che la necessità di trattare in maniera opportuna dati testuali appartiene ormai alla maggior parte dei progetti software, soprattutto grazie all'enorme sviluppo di Internet. In questa tesi è stato presentato un componente estremamente avanzato, che permette di realizzare con facilità soluzioni anche complesse. Per testimoniare la potenza di Lucene posso aggiungere che su questa base è stato costruito un progetto, Nutch (http://lucene.apache.org/nutch), che ha creato un'infrastruttura tecnologica assolutamente in grado di competere con quelle dei più diffusi motori di ricerca commerciali.
Per rimanere nel campo delle soluzioni che lo sviluppatore si trova a dover realizzare più frequentemente segnalo, ad esempio, la possibilità di estrarre da un indice documenti in base alla similitudine: in questa maniera un sistema di commercio elettronico potrà mostrare prodotti simili a quello correntemente visualizzato dall'utente (sulla base delle descrizioni), oppure consultando una banca dati sarà possibile estrarre articoli simili per significato ad un testo scelto dall'utente.

Inoltre è possibile realizzare sistemi di correzione ortografica, simili a quelli presenti negli elaboratori di testi.

Appendice

<specifiche XML>

Tabelle

Tabella delle Query.

Query	Lucene	MySql	Tabelle attraversate
Q1	ID:”article”	Select content.id from content, class_area, class_area_set where class_area.text=”article” and class_area.id= class_area_set. class_area_id and class_area_set.content_id=content.id	3
Q2	Year:”1995”	Select content.id from content where content.year=”1995”	1
Q3	Booktitle:”Modern”	Select content.id from content, title where title.content_id=content.id and title.book_title LIKE “%Modern%”	2
Q4	Title:theory*	Select content.id from content, title where title.title_text LIKE “%theory%” and title.content_id = content.id	2
Q5	Author:kim*	Select content.id from content, person_roles, person_name where person_name.commonName Like “%kim%” and person_name.id= person_roles.person_name_id and person_roles.content_id=content.id	3
Q6	author:bertin*	Select content.id from content, person_roles, person_name where person_name.commonName Like “%bertin%” and person_name.id= person_roles.person_name_id and person_roles.content_id=content.id	3
Q7	year:2007 AND author:Yung	Select content.id from content, person_roles, person_name where content.year=”2007” and person_name.commonName Like “%Yung%” and person_name.id= person_roles.person_name_id and person_roles.content_id=content.id	3
Q8	ID:improceedings AND author:”Yung”	Select content.id from ((((select class_area.id from class_area where class_area.text="book") as class_area left outer join class_area_set on class_area.id= class_area_set.class_area_id) left outer join content on class_area_set.content_id=content.id) left outer join person_roles on person_roles.content_id=content.id) left outer join person_name on	5

		person_name.id= person_roles.person_name_id where person_name.commonName Like "%Yung%";

<codici dei programmi>

6 Bibliografia

6.1.2 Riferimenti bibliografici

Lucene
http://lucene.apache.org

Lucene Wiki
http://wiki.apache.org/jakarta-lucene

E. Hatcher, O. Gospodnetic, Lucene in Action, Manning, 2004

Mailing List general@lucene.apache.org
(archivi su http://mail-archives.apache.org/mod_mbox/lucene-general/)

DBLP Computer Science Bibliography
Il file di dati XML usato per effettuare i test di verifica delle prestazioni fra i due approcci scelti sono stati presi dal link: http://dblp.uni-trier.de/xml/?C=D;O=A , http://dblp.uni-trier.de/ .

Link del convertitore Syntext di file da .dtd a .xml è stato scaricato dal sito:
http://www.w3.org/XML/Schema.

6.1.2.1 Dtd2Xs (Syntext)

Syntext Dtd2Xs v1.3 - Complex&Modularized XML DTD to XML Schema: XML DTD to XML Schema Converter, from Syntext.

"Dtd2Xs allows to convert complex, modularized XML DTDs and DTDs with namespaces to XML Schemas. As an example of Dtd2Xs conversion check out DocBook XML Schema generated fromXML DocBook DTD V4.2, and XSL-FO Schema generated fromXSL-FO DTD."

Changes from last release: Multiple ATTLISTS for the same element now handled correctly; When generating <xs:import namespace="XX">, XX is now namespace URI.

Available for Win32 and Linux: http://www.syntext.com/products/index.htm#Dtd2Xs.

[Editor's note: this appears to be distinct from the other *dtd2xs* in this list of tools.]

http://www.syntext.com/downloads/index.htm#Dtd2Xs

La documentazione e esempi importati circa l'utilizzo di Lucene sono stati raccolti dal sito : http://www.javastaff.com/article.php?story=20070129174128890 .

Articoli correlati : http://www2.mokabyte.it/cms/article.run?articleId=i_NI8-TVY-KIJ-TJU_7f000001_11024688_5b598ab5 .

Per download, licenze e documentazione di Lucene si rimanda al sito ufficiale http://lucene.apache.org/java/docs/index.html .

http://www.opensymphony.com/compass/versions/0.9.0/html/core-searchengine.html